

general_ledger

Generated by Doxygen 1.8.1.2

Sat Jun 7 2014 18:54:03

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	7
3.1	ds_list Struct Reference	7
3.1.1	Detailed Description	7
3.1.2	Field Documentation	7
3.1.2.1	current	8
3.1.2.2	data_destructor	8
3.1.2.3	free_on_delete	8
3.1.2.4	head	8
3.1.2.5	length	8
3.1.2.6	tail	8
3.2	ds_list_element Struct Reference	8
3.2.1	Detailed Description	8
3.2.2	Field Documentation	9
3.2.2.1	data	9
3.2.2.2	next	9
3.2.2.3	previous	9
3.3	ds_map Struct Reference	9
3.3.1	Detailed Description	9
3.3.2	Field Documentation	10
3.3.2.1	hash_size	10
3.3.2.2	lists	10
3.4	ds_map_str Struct Reference	10
3.4.1	Detailed Description	10
3.4.2	Field Documentation	10
3.4.2.1	hash_size	11
3.4.2.2	lists	11

3.5	ds_record Struct Reference	11
3.5.1	Detailed Description	11
3.5.2	Field Documentation	11
3.5.2.1	fields	11
3.6	ds_recordset Struct Reference	12
3.6.1	Detailed Description	12
3.6.2	Field Documentation	12
3.6.2.1	field_lengths	12
3.6.2.2	headers	12
3.6.2.3	num_fields	12
3.6.2.4	records	13
3.7	ds_str Struct Reference	13
3.7.1	Detailed Description	13
3.7.2	Field Documentation	13
3.7.2.1	capacity	13
3.7.2.2	data	13
3.7.2.3	length	13
3.8	ds_vector Struct Reference	13
3.8.1	Detailed Description	13
3.8.2	Field Documentation	14
3.8.2.1	current	14
3.8.2.2	data	14
3.8.2.3	data_destructor	14
3.8.2.4	free_on_delete	14
3.8.2.5	size	14
3.9	kv_pair_node Struct Reference	14
3.9.1	Detailed Description	15
3.9.2	Field Documentation	15
3.9.2.1	key	15
3.9.2.2	key	15
3.9.2.3	next	15
3.9.2.4	value	15
3.9.2.5	value	15
3.10	params Struct Reference	15
3.10.1	Detailed Description	16
3.10.2	Field Documentation	16
3.10.2.1	create	16
3.10.2.2	database	16
3.10.2.3	delete_data	16
3.10.2.4	help	17

3.10.2.5	hostname	17
3.10.2.6	list_entities	17
3.10.2.7	list_users	17
3.10.2.8	password	17
3.10.2.9	sample	17
3.10.2.10	username	17
3.10.2.11	version	17
4	File Documentation	19
4.1	config.c File Reference	19
4.1.1	Detailed Description	20
4.1.2	Macro Definition Documentation	20
4.1.2.1	_XOPEN_SOURCE	20
4.1.3	Function Documentation	20
4.1.3.1	get_cmdline_options	20
4.1.3.2	get_configuration	20
4.1.3.3	params_free	21
4.1.3.4	params_init	21
4.2	config.h File Reference	21
4.2.1	Detailed Description	22
4.2.2	Function Documentation	22
4.2.2.1	get_cmdline_options	22
4.2.2.2	get_configuration	23
4.2.2.3	params_free	23
4.2.2.4	params_init	23
4.3	lib/database/database.h File Reference	23
4.3.1	Detailed Description	24
4.4	lib/database/db_connection.h File Reference	25
4.4.1	Detailed Description	25
4.4.2	Function Documentation	26
4.4.2.1	db_connect	26
4.5	lib/database/db_entities.c File Reference	26
4.5.1	Detailed Description	27
4.5.2	Function Documentation	27
4.5.2.1	db_create_entities_table	27
4.5.2.2	db_drop_entities_table	27
4.5.2.3	db_list_entities_report	27
4.6	lib/database/db_entities.h File Reference	27
4.6.1	Detailed Description	28
4.6.2	Function Documentation	29

4.6.2.1	db_create_entities_table	29
4.6.2.2	db_drop_entities_table	29
4.6.2.3	db_list_entities_report	29
4.7	lib/database/db_internal.h File Reference	29
4.7.1	Detailed Description	30
4.8	lib/database/db_query.h File Reference	30
4.8.1	Detailed Description	31
4.8.2	Function Documentation	31
4.8.2.1	db_execute_query	31
4.9	lib/database/db_reporting.c File Reference	31
4.9.1	Detailed Description	32
4.9.2	Function Documentation	32
4.9.2.1	db_create_report_from_query	32
4.10	lib/database/db_reporting.h File Reference	33
4.10.1	Detailed Description	33
4.10.2	Function Documentation	33
4.10.2.1	db_create_recordset_from_query	33
4.10.2.2	db_create_report_from_query	33
4.11	lib/database/db_sampledata.c File Reference	34
4.11.1	Detailed Description	34
4.12	lib/database/db_sampledata.h File Reference	35
4.12.1	Detailed Description	35
4.13	lib/database/db_sql.h File Reference	35
4.13.1	Detailed Description	36
4.13.2	Function Documentation	36
4.13.2.1	db_create_entities_table_sql	36
4.13.2.2	db_create_users_table_sql	37
4.13.2.3	db_drop_entities_table_sql	37
4.13.2.4	db_drop_users_table_sql	37
4.13.2.5	db_list_entities_report_sql	37
4.13.2.6	db_list_users_report_sql	37
4.14	lib/database/db_structure.c File Reference	37
4.14.1	Detailed Description	38
4.14.2	Function Documentation	38
4.14.2.1	db_create_database_structure	38
4.14.2.2	db_delete_database_structure	39
4.15	lib/database/db_structure.h File Reference	39
4.15.1	Detailed Description	39
4.15.2	Function Documentation	40
4.15.2.1	db_create_database_structure	40

4.15.2.2	db_delete_database_structure	40
4.16	lib/database/db_users.c File Reference	40
4.16.1	Detailed Description	41
4.16.2	Function Documentation	41
4.16.2.1	db_create_users_table	41
4.16.2.2	db_drop_users_table	41
4.16.2.3	db_list_users_report	41
4.17	lib/database/db_users.h File Reference	42
4.17.1	Detailed Description	42
4.17.2	Function Documentation	43
4.17.2.1	db_create_users_table	43
4.17.2.2	db_drop_users_table	43
4.17.2.3	db_list_users_report	43
4.18	lib/database/dummy/db_dummy_create_entities_table_sql.c File Reference	43
4.18.1	Detailed Description	43
4.18.2	Function Documentation	44
4.18.2.1	db_create_entities_table_sql	44
4.19	lib/database/dummy/db_dummy_create_users_table_sql.c File Reference	44
4.19.1	Detailed Description	44
4.19.2	Function Documentation	44
4.19.2.1	db_create_users_table_sql	44
4.20	lib/database/dummy/db_dummy_drop_entities_table_sql.c File Reference	44
4.20.1	Detailed Description	45
4.20.2	Function Documentation	45
4.20.2.1	db_drop_entities_table_sql	45
4.21	lib/database/dummy/db_dummy_drop_users_table_sql.c File Reference	45
4.21.1	Detailed Description	45
4.21.2	Function Documentation	45
4.21.2.1	db_drop_users_table_sql	45
4.22	lib/database/dummy/db_dummy_general.c File Reference	46
4.22.1	Detailed Description	46
4.22.2	Macro Definition Documentation	47
4.22.2.1	_XOPEN_SOURCE	47
4.22.3	Function Documentation	47
4.22.3.1	db_connect	47
4.22.3.2	db_create_recordset_from_query	47
4.22.3.3	db_execute_query	47
4.23	lib/database/dummy/db_dummy_list_entities_report_sql.c File Reference	48
4.23.1	Detailed Description	48
4.23.2	Function Documentation	48

4.23.2.1 db_list_entities_report_sql	48
4.24 lib/database/dummy/db_dummy_list_users_report_sql.c File Reference	48
4.24.1 Detailed Description	48
4.24.2 Function Documentation	49
4.24.2.1 db_list_users_report_sql	49
4.25 lib/database/mysql/db_mysql_create_entities_table_sql.c File Reference	49
4.25.1 Detailed Description	49
4.25.2 Function Documentation	49
4.25.2.1 db_create_entities_table_sql	49
4.26 lib/database/mysql/db_mysql_create_users_table_sql.c File Reference	49
4.26.1 Detailed Description	50
4.26.2 Function Documentation	50
4.26.2.1 db_create_users_table_sql	50
4.27 lib/database/mysql/db_mysql_drop_entities_table_sql.c File Reference	50
4.27.1 Detailed Description	50
4.27.2 Function Documentation	51
4.27.2.1 db_drop_entities_table_sql	51
4.28 lib/database/mysql/db_mysql_drop_users_table_sql.c File Reference	51
4.28.1 Detailed Description	51
4.28.2 Function Documentation	51
4.28.2.1 db_drop_users_table_sql	51
4.29 lib/database/mysql/db_mysql_general.c File Reference	51
4.29.1 Detailed Description	52
4.29.2 Function Documentation	53
4.29.2.1 db_connect	53
4.29.2.2 db_create_recordset_from_query	53
4.29.2.3 db_execute_query	53
4.29.3 Variable Documentation	53
4.29.3.1 conn_mss	53
4.29.3.2 main_mss	53
4.30 lib/database/mysql/db_mysql_list_entities_report_sql.c File Reference	54
4.30.1 Detailed Description	54
4.30.2 Function Documentation	54
4.30.2.1 db_list_entities_report_sql	54
4.31 lib/database/mysql/db_mysql_list_users_report_sql.c File Reference	54
4.31.1 Detailed Description	54
4.31.2 Function Documentation	55
4.31.2.1 db_list_users_report_sql	55
4.32 lib/datastruct/data_structures.h File Reference	55
4.32.1 Detailed Description	56

4.33 lib/datastruct/ds_list.c File Reference	56
4.33.1 Detailed Description	57
4.33.2 Function Documentation	57
4.33.2.1 ds_list_append	57
4.33.2.2 ds_list_create	58
4.33.2.3 ds_list_destroy	58
4.33.2.4 ds_list_destructor	58
4.33.2.5 ds_list_element	58
4.33.2.6 ds_list_get_next_data	58
4.33.2.7 ds_list_get_prev_data	59
4.33.2.8 ds_list_is_empty	59
4.33.2.9 ds_list_length	59
4.33.2.10 ds_list_remove_all	59
4.33.2.11 ds_list_remove_tail	60
4.33.2.12 ds_list_seek_end	60
4.33.2.13 ds_list_seek_start	60
4.34 lib/datastruct/ds_list.h File Reference	60
4.34.1 Detailed Description	61
4.34.2 Typedef Documentation	62
4.34.2.1 ds_list	62
4.34.3 Function Documentation	62
4.34.3.1 ds_list_append	62
4.34.3.2 ds_list_create	62
4.34.3.3 ds_list_destroy	62
4.34.3.4 ds_list_destructor	63
4.34.3.5 ds_list_element	63
4.34.3.6 ds_list_get_next_data	63
4.34.3.7 ds_list_get_prev_data	63
4.34.3.8 ds_list_is_empty	64
4.34.3.9 ds_list_length	64
4.34.3.10 ds_list_remove_all	64
4.34.3.11 ds_list_remove_tail	64
4.34.3.12 ds_list_seek_end	64
4.34.3.13 ds_list_seek_start	64
4.35 lib/datastruct/ds_map.c File Reference	65
4.35.1 Detailed Description	66
4.35.2 Function Documentation	66
4.35.2.1 ds_map_destroy	66
4.35.2.2 ds_map_get_value	66
4.35.2.3 ds_map_init	66

4.35.2.4	ds_map_insert	67
4.35.2.5	ds_map_print_all	67
4.36	lib/datastruct/ds_map.h File Reference	67
4.36.1	Detailed Description	68
4.36.2	Typedef Documentation	68
4.36.2.1	ds_map	68
4.36.3	Function Documentation	68
4.36.3.1	ds_map_destroy	68
4.36.3.2	ds_map_get_value	69
4.36.3.3	ds_map_init	69
4.36.3.4	ds_map_insert	69
4.36.3.5	ds_map_print_all	69
4.37	lib/datastruct/ds_map_str.c File Reference	70
4.37.1	Detailed Description	70
4.37.2	Function Documentation	71
4.37.2.1	ds_map_str_destroy	71
4.37.2.2	ds_map_str_get_value	71
4.37.2.3	ds_map_str_init	71
4.37.2.4	ds_map_str_insert	71
4.38	lib/datastruct/ds_map_str.h File Reference	72
4.38.1	Detailed Description	73
4.38.2	Typedef Documentation	73
4.38.2.1	ds_map_str	73
4.38.3	Function Documentation	73
4.38.3.1	ds_map_str_destroy	73
4.38.3.2	ds_map_str_get_value	73
4.38.3.3	ds_map_str_init	73
4.38.3.4	ds_map_str_insert	74
4.39	lib/datastruct/ds_record.c File Reference	74
4.39.1	Detailed Description	75
4.39.2	Function Documentation	75
4.39.2.1	ds_record_clear	75
4.39.2.2	ds_record_create	75
4.39.2.3	ds_record_destroy	76
4.39.2.4	ds_record_destructor	76
4.39.2.5	ds_record_get_field	76
4.39.2.6	ds_record_get_next_data	76
4.39.2.7	ds_record_make_delim_string	76
4.39.2.8	ds_record_make_values_string	77
4.39.2.9	ds_record_seek_start	77

4.39.2.10	ds_record_set_field	77
4.39.2.11	ds_record_size	77
4.39.2.12	ds_record_tokenize	78
4.40	lib/datastruct/ds_record.h File Reference	78
4.40.1	Detailed Description	79
4.40.2	Typedef Documentation	79
4.40.2.1	ds_record	79
4.40.3	Function Documentation	79
4.40.3.1	ds_record_clear	79
4.40.3.2	ds_record_create	80
4.40.3.3	ds_record_destroy	80
4.40.3.4	ds_record_destructor	80
4.40.3.5	ds_record_get_field	80
4.40.3.6	ds_record_get_next_data	80
4.40.3.7	ds_record_make_delim_string	81
4.40.3.8	ds_record_make_values_string	81
4.40.3.9	ds_record_seek_start	81
4.40.3.10	ds_record_set_field	81
4.40.3.11	ds_record_size	82
4.40.3.12	ds_record_tokenize	82
4.41	lib/datastruct/ds_recordset.c File Reference	82
4.41.1	Detailed Description	83
4.41.2	Function Documentation	83
4.41.2.1	ds_recordset_add_record	83
4.41.2.2	ds_recordset_create	84
4.41.2.3	ds_recordset_destroy	84
4.41.2.4	ds_recordset_get_next_insert_query	84
4.41.2.5	ds_recordset_get_text_report	84
4.41.2.6	ds_recordset_next_record	84
4.41.2.7	ds_recordset_num_fields	85
4.41.2.8	ds_recordset_num_records	85
4.41.2.9	ds_recordset_seek_start	85
4.41.2.10	ds_recordset_set_headers	85
4.42	lib/datastruct/ds_recordset.h File Reference	86
4.42.1	Detailed Description	87
4.42.2	Typedef Documentation	87
4.42.2.1	ds_recordset	87
4.42.3	Function Documentation	87
4.42.3.1	ds_recordset_add_record	87
4.42.3.2	ds_recordset_create	88

4.42.3.3	ds_recordset_destroy	88
4.42.3.4	ds_recordset_get_next_insert_query	88
4.42.3.5	ds_recordset_get_text_report	88
4.42.3.6	ds_recordset_next_record	88
4.42.3.7	ds_recordset_num_fields	89
4.42.3.8	ds_recordset_num_records	89
4.42.3.9	ds_recordset_seek_start	89
4.42.3.10	ds_recordset_set_headers	89
4.43	lib/datastruct/ds_str.c File Reference	90
4.43.1	Detailed Description	92
4.43.2	Function Documentation	92
4.43.2.1	ds_str_assign	92
4.43.2.2	ds_str_assign_cstr	92
4.43.2.3	ds_str_char_at_index	92
4.43.2.4	ds_str_clear	93
4.43.2.5	ds_str_compare	93
4.43.2.6	ds_str_compare_cstr	93
4.43.2.7	ds_str_concat	93
4.43.2.8	ds_str_concat_cstr	93
4.43.2.9	ds_str_create	94
4.43.2.10	ds_str_create_direct	94
4.43.2.11	ds_str_create_sprintf	94
4.43.2.12	ds_str_cstr	95
4.43.2.13	ds_str_decorate	95
4.43.2.14	ds_str_destroy	95
4.43.2.15	ds_str_destructor	95
4.43.2.16	ds_str_doubleval	95
4.43.2.17	ds_str_dup	96
4.43.2.18	ds_str_getline	96
4.43.2.19	ds_str_hash	96
4.43.2.20	ds_str_intval	96
4.43.2.21	ds_str_is_empty	97
4.43.2.22	ds_str_length	97
4.43.2.23	ds_str_size_to_fit	97
4.43.2.24	ds_str_split	97
4.43.2.25	ds_str_strchr	98
4.43.2.26	ds_str_substr_left	98
4.43.2.27	ds_str_substr_right	98
4.43.2.28	ds_str_trim	98
4.43.2.29	ds_str_trim_leading	99

4.43.2.30 ds_str_trim_trailing	99
4.43.2.31 ds_str_trunc	99
4.44 lib/datastruct/ds_str.h File Reference	99
4.44.1 Detailed Description	101
4.44.2 Typedef Documentation	101
4.44.2.1 ds_str	101
4.44.3 Function Documentation	102
4.44.3.1 ds_str_assign	102
4.44.3.2 ds_str_assign_cstr	102
4.44.3.3 ds_str_char_at_index	102
4.44.3.4 ds_str_clear	102
4.44.3.5 ds_str_compare	102
4.44.3.6 ds_str_compare_cstr	103
4.44.3.7 ds_str_concat	103
4.44.3.8 ds_str_concat_cstr	103
4.44.3.9 ds_str_create	103
4.44.3.10 ds_str_create_direct	104
4.44.3.11 ds_str_create_sprintf	104
4.44.3.12 ds_str_cstr	104
4.44.3.13 ds_str_decorate	105
4.44.3.14 ds_str_destroy	105
4.44.3.15 ds_str_destructor	105
4.44.3.16 ds_str_doubleval	105
4.44.3.17 ds_str_dup	105
4.44.3.18 ds_str_getline	106
4.44.3.19 ds_str_hash	106
4.44.3.20 ds_str_intval	106
4.44.3.21 ds_str_is_empty	107
4.44.3.22 ds_str_length	107
4.44.3.23 ds_str_size_to_fit	107
4.44.3.24 ds_str_split	107
4.44.3.25 ds_str_strchr	107
4.44.3.26 ds_str_substr_left	108
4.44.3.27 ds_str_substr_right	108
4.44.3.28 ds_str_trim	108
4.44.3.29 ds_str_trim_leading	108
4.44.3.30 ds_str_trim_trailing	109
4.44.3.31 ds_str_trunc	109
4.45 lib/datastruct/ds_vector.c File Reference	109
4.45.1 Detailed Description	110

4.45.2	Function Documentation	110
4.45.2.1	ds_vector_clear	110
4.45.2.2	ds_vector_create	111
4.45.2.3	ds_vector_destroy	111
4.45.2.4	ds_vector_destructor	111
4.45.2.5	ds_vector_element	111
4.45.2.6	ds_vector_get_next_data	111
4.45.2.7	ds_vector_seek_start	112
4.45.2.8	ds_vector_set	112
4.45.2.9	ds_vector_size	112
4.46	lib/datastruct/ds_vector.h File Reference	113
4.46.1	Detailed Description	114
4.46.2	Typedef Documentation	114
4.46.2.1	ds_vector	114
4.46.3	Function Documentation	114
4.46.3.1	ds_vector_clear	114
4.46.3.2	ds_vector_create	114
4.46.3.3	ds_vector_destroy	115
4.46.3.4	ds_vector_destructor	115
4.46.3.5	ds_vector_element	115
4.46.3.6	ds_vector_get_next_data	115
4.46.3.7	ds_vector_seek_start	116
4.46.3.8	ds_vector_set	116
4.46.3.9	ds_vector_size	116
4.47	lib/file_ops/config_file_read.c File Reference	116
4.47.1	Detailed Description	117
4.47.2	Macro Definition Documentation	117
4.47.2.1	CONFIG_MAP_SIZE	117
4.47.2.2	MAX_BUFFER_SIZE	118
4.47.3	Function Documentation	118
4.47.3.1	config_file_free	118
4.47.3.2	config_file_read	118
4.47.3.3	config_file_value	118
4.48	lib/file_ops/config_file_read.h File Reference	118
4.48.1	Detailed Description	120
4.48.2	Macro Definition Documentation	120
4.48.2.1	CONFIG_FILE_MALFORMED_FILE	120
4.48.2.2	CONFIG_FILE_NO_FILE	120
4.48.2.3	CONFIG_FILE_OK	120
4.48.3	Function Documentation	120

4.48.3.1	config_file_free	120
4.48.3.2	config_file_read	120
4.48.3.3	config_file_value	121
4.49	lib/file_ops/delim_file_read.c File Reference	121
4.49.1	Detailed Description	122
4.49.2	Macro Definition Documentation	122
4.49.2.1	MAX_LINE_SIZE	122
4.49.3	Function Documentation	122
4.49.3.1	delim_file_read	122
4.50	lib/file_ops/delim_file_read.h File Reference	122
4.50.1	Detailed Description	124
4.50.2	Function Documentation	124
4.50.2.1	delim_file_read	124
4.51	lib/file_ops/file_ops.h File Reference	124
4.51.1	Detailed Description	125
4.52	lib/gl_general/gl_errors.c File Reference	126
4.52.1	Detailed Description	126
4.52.2	Function Documentation	126
4.52.2.1	gl_error_quit	126
4.53	lib/gl_general/gl_errors.h File Reference	127
4.53.1	Detailed Description	127
4.53.2	Function Documentation	127
4.53.2.1	gl_error_quit	127
4.54	lib/gl_general/gl_general.h File Reference	127
4.54.1	Detailed Description	128
4.55	lib/gl_general/gl_logging.c File Reference	128
4.55.1	Detailed Description	129
4.55.2	Function Documentation	129
4.55.2.1	gl_log_msg	129
4.55.2.2	gl_set_logging	130
4.56	lib/gl_general/gl_logging.h File Reference	130
4.56.1	Detailed Description	130
4.56.2	Function Documentation	131
4.56.2.1	gl_log_msg	131
4.56.2.2	gl_set_logging	131
4.57	main.c File Reference	131
4.57.1	Detailed Description	132
4.57.2	Function Documentation	132
4.57.2.1	login	132
4.57.2.2	main	133

4.57.2.3	print_help_message	133
4.57.2.4	print_usage_message	133
4.57.2.5	print_version_message	133
4.57.2.6	test_functionality	133

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

ds_list	7
ds_list_element	8
ds_map	9
ds_map_str	10
ds_record	11
ds_recordset	12
ds_str	13
ds_vector	13
kv_pair_node	14
params	15

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

config.c	Implementation of program configuration functionality	19
config.h	Interface to program configuration functionality	21
main.c	Main function for general_ledger	131
lib/database/database.h	User interface to database functionality	23
lib/database/db_connection.h	Interface to database connection functionality	25
lib/database/db_entities.c	Implementation of entities functionality	26
lib/database/db_entities.h	Interface to entities functionality	27
lib/database/db_internal.h	Internal library interface to database functionality	29
lib/database/db_query.h	Interface to database query functionality	30
lib/database/db_reporting.c	Implementation of database reporting functionality	31
lib/database/db_reporting.h	Interface to database reporting functionality	33
lib/database/db_sampledata.c	Implementation of database sample data functionality	34
lib/database/db_sampledata.h	Interface to database sample data functionality	35
lib/database/db_sql.h	Interface to database specific SQL strings	35
lib/database/db_structure.c	Implementation of database structure functionality	37
lib/database/db_structure.h	Interface to database structure functionality	39
lib/database/db_users.c	Implementation of users functionality	40
lib/database/db_users.h	Interface to users functionality	42
lib/database/dummy/db_dummy_create_entities_table_sql.c	Returns dummy SQL query to create entities table	43

lib/database/dummy/db_dummy_create_users_table_sql.c	Returns dummy SQL query to create users table	44
lib/database/dummy/db_dummy_drop_entities_table_sql.c	Returns dummy SQL query to drop entities table	44
lib/database/dummy/db_dummy_drop_users_table_sql.c	Returns dummy SQL query to drop users table	45
lib/database/dummy/db_dummy_general.c	Implementation of dummy database functionality	46
lib/database/dummy/db_dummy_list_entities_report_sql.c	Returns dummy SQL query to create list entities report	48
lib/database/dummy/db_dummy_list_users_report_sql.c	Returns dummy SQL query to create list users report	48
lib/database/mysql/db_mysql_create_entities_table_sql.c	Returns MYSQL SQL query to create entities table	49
lib/database/mysql/db_mysql_create_users_table_sql.c	Returns MYSQL SQL query to create users table	49
lib/database/mysql/db_mysql_drop_entities_table_sql.c	Returns MYSQL SQL query to drop entities table	50
lib/database/mysql/db_mysql_drop_users_table_sql.c	Returns MYSQL SQL query to drop users table	51
lib/database/mysql/db_mysql_general.c	Implementation of MYSQL database functionality	51
lib/database/mysql/db_mysql_list_entities_report_sql.c	Returns MYSQL SQL query to create list entities report	54
lib/database/mysql/db_mysql_list_users_report_sql.c	Returns MYSQL SQL query to create list users report	54
lib/datastruct/data_structures.h	Interface to data structures	55
lib/datastruct/ds_list.c	Implementation of generic doubly-linked list data structure	56
lib/datastruct/ds_list.h	Interface to generic doubly-linked list data structure	60
lib/datastruct/ds_map.c	Implementation of string-string hash map data structure	65
lib/datastruct/ds_map.h	Interface to string-string hash map data structure	67
lib/datastruct/ds_map_str.c	Implementation of string-string hash map data structure	70
lib/datastruct/ds_map_str.h	Interface to string-string hash map data structure	72
lib/datastruct/ds_record.c	Implementation of record database structure	74
lib/datastruct/ds_record.h	Interface to record data structure	78
lib/datastruct/ds_recordset.c	Implementation of query result set structure	82
lib/datastruct/ds_recordset.h	Interface to record set structure	86
lib/datastruct/ds_str.c	Implementation of string data structure	90
lib/datastruct/ds_str.h	Interface to string data structure	99
lib/datastruct/ds_vector.c	Implementation of generic doubly-linked vector data structure	109
lib/datastruct/ds_vector.h	Interface to generic doubly-linked vector data structure	113
lib/file_ops/config_file_read.c	Implementation of configuration file reading functionality	116

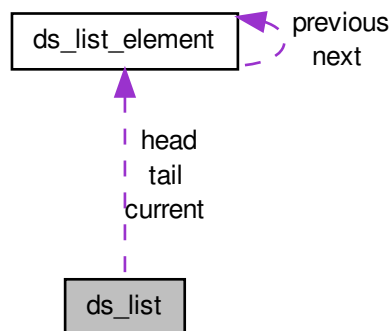
lib/file_ops/ config_file_read.h	
Interface to configuration file reading functionality	118
lib/file_ops/ delim_file_read.c	
Implementation of delimited file reading functionality	121
lib/file_ops/ delim_file_read.h	
Interface to delimited file reading functionality	122
lib/file_ops/ file_ops.h	
User interface to file operations functionality	124
lib/gl_general/ gl_errors.c	
Implementation of error functionality	126
lib/gl_general/ gl_errors.h	
Interface to error functionality	127
lib/gl_general/ gl_general.h	
User interface to logging and error functionality	127
lib/gl_general/ gl_logging.c	
Implementation of logging functionality	128
lib/gl_general/ gl_logging.h	
Interface to logging functionality	130

Chapter 3

Data Structure Documentation

3.1 ds_list Struct Reference

Collaboration diagram for ds_list:



Data Fields

- `size_t` `length`
- `bool` `free_on_delete`
- `struct ds_list_element *` `head`
- `struct ds_list_element *` `tail`
- `struct ds_list_element *` `current`
- `void(* data_destructor)(void *)`

3.1.1 Detailed Description

List data structure

3.1.2 Field Documentation

3.1.2.1 struct ds_list_element* ds_list::current

Pointer to current element

3.1.2.2 void(* ds_list::data_destructor)(void *)

Data destructor function

3.1.2.3 bool ds_list::free_on_delete

'Free on delete' flag

3.1.2.4 struct ds_list_element* ds_list::head

Pointer to head element

3.1.2.5 size_t ds_list::length

Length of list

3.1.2.6 struct ds_list_element* ds_list::tail

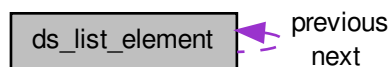
Pointer to tail element

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_list.c](#)

3.2 ds_list_element Struct Reference

Collaboration diagram for ds_list_element:

**Data Fields**

- void * [data](#)
- struct [ds_list_element](#) * [previous](#)
- struct [ds_list_element](#) * [next](#)

3.2.1 Detailed Description

List element data structure

3.2.2 Field Documentation

3.2.2.1 void* ds_list_element::data

Pointer to data

3.2.2.2 struct ds_list_element* ds_list_element::next

Pointer to next element

3.2.2.3 struct ds_list_element* ds_list_element::previous

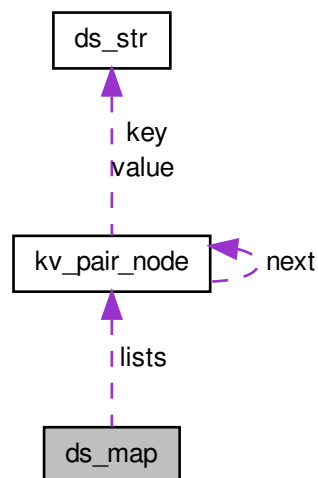
Pointer to previous element

The documentation for this struct was generated from the following file:

- lib/datastruct/[ds_list.c](#)

3.3 ds_map Struct Reference

Collaboration diagram for ds_map:



Data Fields

- struct [kv_pair_node](#) ** `lists`
- size_t [hash_size](#)

3.3.1 Detailed Description

Structure to hold a hash map

3.3.2 Field Documentation

3.3.2.1 `size_t ds_map::hash_size`

Size of array of lists

3.3.2.2 `struct kv_pair_node** ds_map::lists`

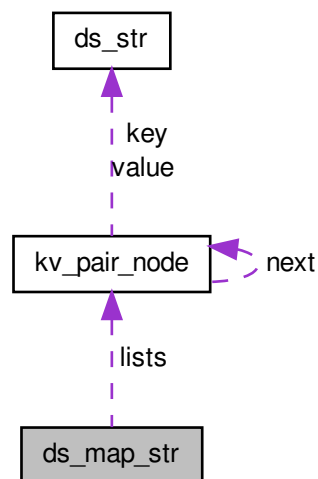
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_map.c](#)

3.4 `ds_map_str` Struct Reference

Collaboration diagram for `ds_map_str`:



Data Fields

- struct [kv_pair_node](#) ** `lists`
- `size_t` [hash_size](#)

3.4.1 Detailed Description

Structure to hold a hash map

3.4.2 Field Documentation

3.4.2.1 `size_t ds_map_str::hash_size`

Size of array of lists

3.4.2.2 `struct kv_pair_node** ds_map_str::lists`

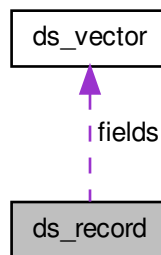
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_map_str.c](#)

3.5 ds_record Struct Reference

Collaboration diagram for ds_record:



Data Fields

- struct [ds_vector](#) * [fields](#)

3.5.1 Detailed Description

Vector data structure

3.5.2 Field Documentation

3.5.2.1 `struct ds_vector* ds_record::fields`

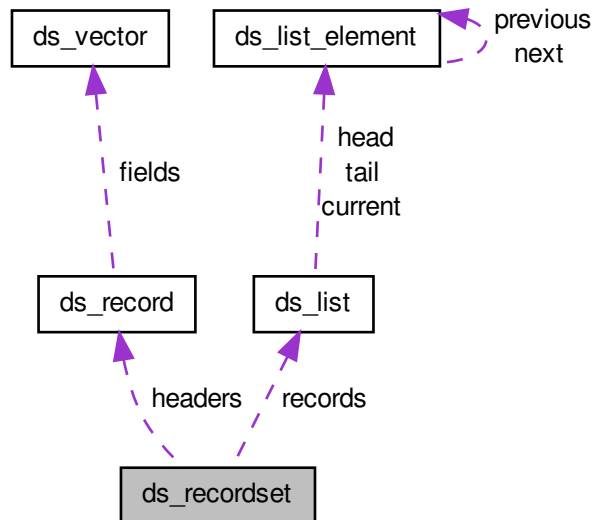
Vector of fields

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_record.c](#)

3.6 ds_recordset Struct Reference

Collaboration diagram for ds_recordset:



Data Fields

- `size_t num_fields`
- `size_t * field_lengths`
- `ds_record headers`
- `ds_list records`

3.6.1 Detailed Description

Result set structure

3.6.2 Field Documentation

3.6.2.1 `size_t * ds_recordset::field_lengths`

Lengths of the longest fields

3.6.2.2 `ds_record ds_recordset::headers`

A list of field headers

3.6.2.3 `size_t ds_recordset::num_fields`

The number of fields in a record

3.6.2.4 ds_list ds_recordset::records

A list of records

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_recordset.c](#)

3.7 ds_str Struct Reference

Data Fields

- `char *` [data](#)
- `size_t` [length](#)
- `size_t` [capacity](#)

3.7.1 Detailed Description

Structure to contain string

3.7.2 Field Documentation

3.7.2.1 size_t ds_str::capacity

The size of the `data` buffer

3.7.2.2 char* ds_str::data

The data in C-style string format

3.7.2.3 size_t ds_str::length

The length of the string

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_str.c](#)

3.8 ds_vector Struct Reference

Data Fields

- `size_t` [size](#)
- `size_t` [current](#)
- `bool` [free_on_delete](#)
- `void **` [data](#)
- `void(*` [data_destructor](#) `)(void *)`

3.8.1 Detailed Description

Vector data structure

3.8.2 Field Documentation

3.8.2.1 `size_t ds_vector::current`

Current position

3.8.2.2 `void** ds_vector::data`

Data array

3.8.2.3 `void(* ds_vector::data_destructor)(void *)`

Data destructor function

3.8.2.4 `bool ds_vector::free_on_delete`

'Free on delete' flag

3.8.2.5 `size_t ds_vector::size`

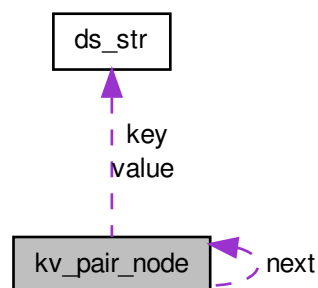
Size of vector

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_vector.c](#)

3.9 kv_pair_node Struct Reference

Collaboration diagram for kv_pair_node:



Data Fields

- `char * key`
- `char * value`
- `struct kv_pair_node * next`

- [ds_str key](#)
- [ds_str value](#)

3.9.1 Detailed Description

Structure to hold a key-value pair node

3.9.2 Field Documentation

3.9.2.1 `ds_str kv_pair_node::key`

A pointer to the key

3.9.2.2 `char* kv_pair_node::key`

A pointer to the key

3.9.2.3 `struct kv_pair_node * kv_pair_node::next`

A pointer to the next node

3.9.2.4 `ds_str kv_pair_node::value`

A pointer to the value

3.9.2.5 `char* kv_pair_node::value`

A pointer to the value

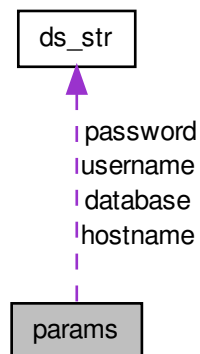
The documentation for this struct was generated from the following files:

- [lib/datastruct/ds_map.c](#)
- [lib/datastruct/ds_map_str.c](#)

3.10 params Struct Reference

```
#include <config.h>
```

Collaboration diagram for params:



Data Fields

- [ds_str hostname](#)
- [ds_str database](#)
- [ds_str username](#)
- [ds_str password](#)
- [bool help](#)
- [bool version](#)
- [bool create](#)
- [bool delete_data](#)
- [bool sample](#)
- [bool list_users](#)
- [bool list_entities](#)

3.10.1 Detailed Description

Structure to hold program parameters

3.10.2 Field Documentation

3.10.2.1 `bool params::create`

Create structure option set

3.10.2.2 `ds_str params::database`

Database name

3.10.2.3 `bool params::delete_data`

Delete structure option set

3.10.2.4 bool params::help

Help option set

3.10.2.5 ds_str params::hostname

Database hostname

3.10.2.6 bool params::list_entities

List entities option set

3.10.2.7 bool params::list_users

List users option set

3.10.2.8 ds_str params::password

Password for database access

3.10.2.9 bool params::sample

Load sample data option set

3.10.2.10 ds_str params::username

Username for database access

3.10.2.11 bool params::version

Version option set

The documentation for this struct was generated from the following file:

- [config.h](#)

Chapter 4

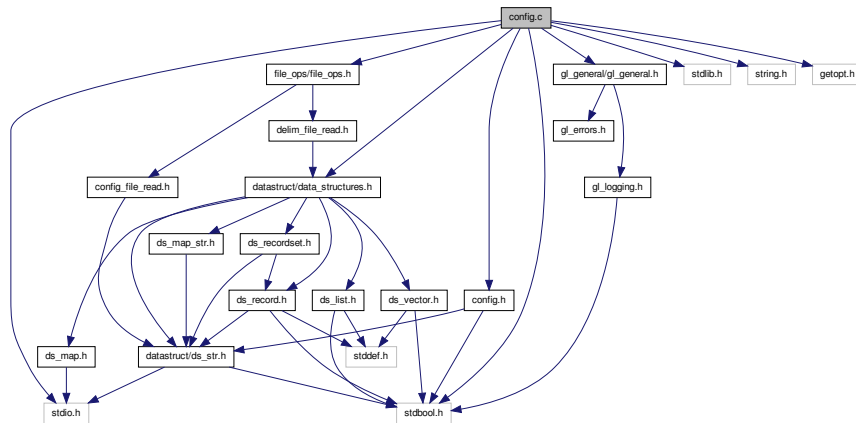
File Documentation

4.1 config.c File Reference

Implementation of program configuration functionality.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "config.h"
#include "file_ops/file_ops.h"
#include "datastruct/data_structures.h"
#include "gl_general/gl_general.h"
```

Include dependency graph for config.c:



Macros

- `#define _XOPEN_SOURCE 500`

Functions

- `struct params * params_init (void)`
Initializes a parameters structure.

- void `params_free` (struct `params` *`params`)
Frees a parameter structure.
- bool `get_configuration` (struct `params` *`params`)
Gets parameters from a configuration file.
- bool `get_cmdline_options` (int `argc`, char **`argv`, struct `params` *`params`)
Gets parameters from the command line.

4.1.1 Detailed Description

Implementation of program configuration functionality. Gets program configuration options from the command line and/or a configuration file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.1.2 Macro Definition Documentation

4.1.2.1 `#define XOPEN.SOURCE 500`

UNIX feature test macro

4.1.3 Function Documentation

4.1.3.1 bool `get_cmdline_options` (int *argc*, char ** *argv*, struct `params` * *params*)

Gets parameters from the command line.

Parameters

<i>argc</i>	<code>argc</code> as passed to <code>main()</code> .
<i>argv</i>	<code>argv</code> as passed to <code>main()</code> .
<i>params</i>	A pointer to a parameters structure to populate.

Returns

`false` if an unrecognized command line option was specified, `true` otherwise.

4.1.3.2 bool `get_configuration` (struct `params` * *params*)

Gets parameters from a configuration file.

Parameters

<i>params</i>	A pointer to a parameters structure to populate.
---------------	--

Returns

`true` on success, `false` otherwise.

4.1.3.3 void params_free (struct params * params)

Frees a parameter structure.

Parameters

<i>params</i>	A pointer to the structure to free.
---------------	-------------------------------------

4.1.3.4 struct params* params_init (void) [read]

Initializes a parameters structure.

Returns

An initialized parameters structure.

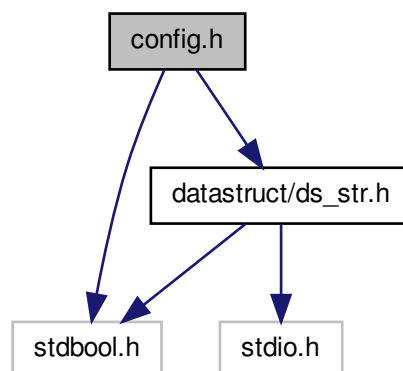
4.2 config.h File Reference

Interface to program configuration functionality.

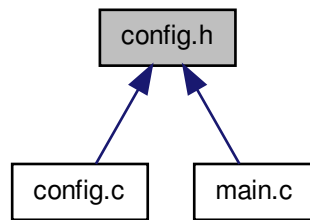
```
#include <stdbool.h>
```

```
#include "datastruct/ds_str.h"
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [params](#)

Functions

- struct [params](#) * [params_init](#) (void)
Initializes a parameters structure.
- void [params_free](#) (struct [params](#) *[params](#))
Frees a parameter structure.
- bool [get_configuration](#) (struct [params](#) *[params](#))
Gets parameters from a configuration file.
- bool [get_cmdline_options](#) (int argc, char **argv, struct [params](#) *[params](#))
Gets parameters from the command line.

4.2.1 Detailed Description

Interface to program configuration functionality. Gets program configuration options from the command line and/or a configuration file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.2.2 Function Documentation

4.2.2.1 bool [get_cmdline_options](#) (int *argc*, char ** *argv*, struct [params](#) * *params*)

Gets parameters from the command line.

Parameters

<i>argc</i>	<code>argc</code> as passed to <code>main()</code> .
<i>argv</i>	<code>argv</code> as passed to <code>main()</code> .
<i>params</i>	A pointer to a parameters structure to populate.

Returns

`false` if an unrecognized command line option was specified, `true` otherwise.

4.2.2.2 bool get_configuration (struct params * params)

Gets parameters from a configuration file.

Parameters

<i>params</i>	A pointer to a parameters structure to populate.
---------------	--

Returns

`true` on success, `false` otherwise.

4.2.2.3 void params_free (struct params * params)

Frees a parameter structure.

Parameters

<i>params</i>	A pointer to the structure to free.
---------------	-------------------------------------

4.2.2.4 struct params* params_init (void) [read]

Initializes a parameters structure.

Returns

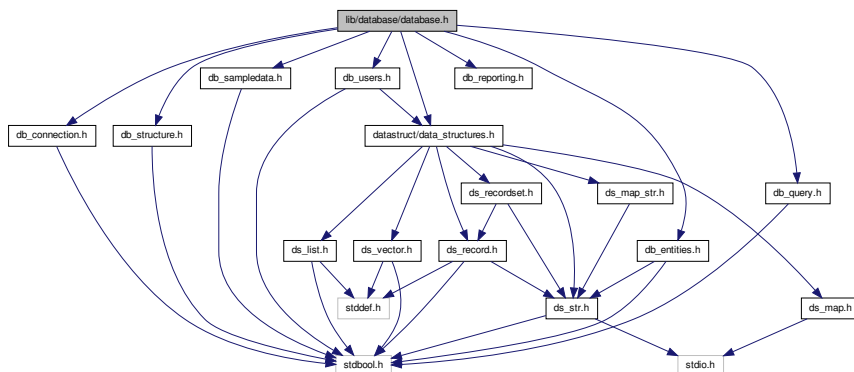
An initialized parameters structure.

4.3 lib/database/database.h File Reference

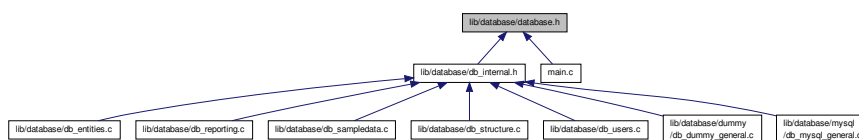
User interface to database functionality.

```
#include "datastruct/data_structures.h"
#include "db_connection.h"
#include "db_structure.h"
#include "db_query.h"
#include "db_sampledata.h"
#include "db_reporting.h"
#include "db_users.h"
#include "db_entities.h"
```

Include dependency graph for database.h:



This graph shows which files directly or indirectly include this file:



4.3.1 Detailed Description

User interface to database functionality.

Author

Paul Griffiths

Copyright

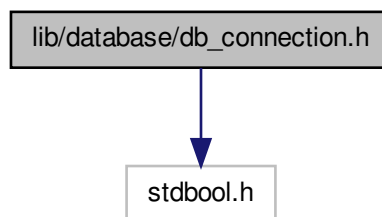
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.4 lib/database/db_connection.h File Reference

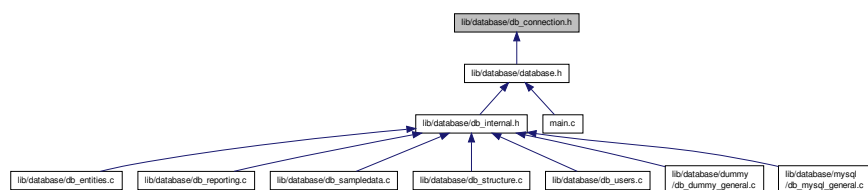
Interface to database connection functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_connection.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool **db_connect** (const char *host, const char *database, const char *username, const char *password)
Connects to a database.
- void **db_close** (void)
Disconnects from a database.

4.4.1 Detailed Description

Interface to database connection functionality. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.4.2 Function Documentation

4.4.2.1 `bool db_connect (const char * host, const char * database, const char * username, const char * password)`

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

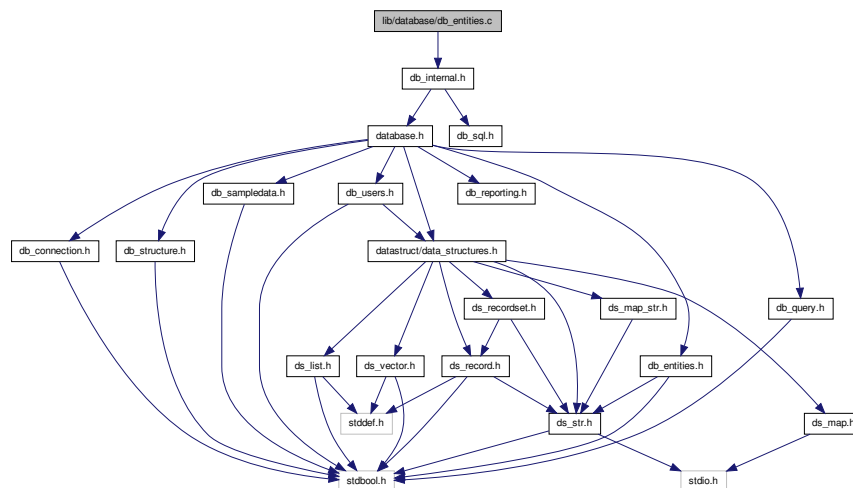
`true` if the connection was successfully made, `false` otherwise.

4.5 `lib/database/db_entities.c` File Reference

Implementation of entities functionality.

```
#include "db_internal.h"
```

Include dependency graph for `db_entities.c`:



Functions

- `bool db_create_entities_table (void)`
Creates the entities table in the database.
- `bool db_drop_entities_table (void)`
Drops the entities table in the database.
- `ds_str db_list_entities_report (void)`
Creates a report listing all entities.

4.5.1 Detailed Description

Implementation of entities functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.5.2 Function Documentation

4.5.2.1 `bool db_create_entities_table (void)`

Creates the entities table in the database.

Returns

`true` on success, `false` on failure.

4.5.2.2 `bool db_drop_entities_table (void)`

Drops the entities table in the database.

Returns

`true` on success, `false` on failure.

4.5.2.3 `ds_str db_list_entities_report (void)`

Creates a report listing all entities.

Returns

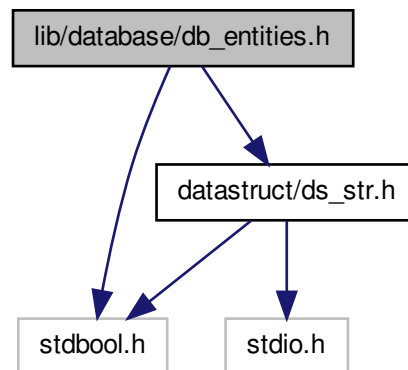
A `ds_str` containing the report.

4.6 lib/database/db_entities.h File Reference

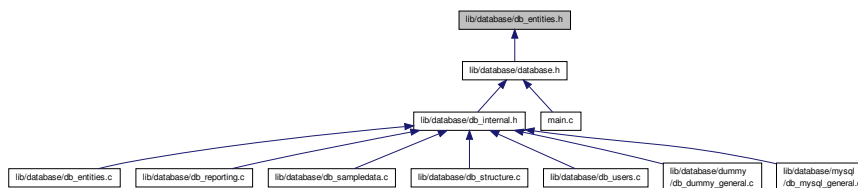
Interface to entities functionality.

```
#include <stdbool.h>
#include "datastruct/ds_str.h"
```

Include dependency graph for `db_entities.h`:



This graph shows which files directly or indirectly include this file:



Functions

- bool `db_create_entities_table` (void)
Creates the entities table in the database.
- bool `db_drop_entities_table` (void)
Drops the entities table in the database.
- `ds_str db_list_entities_report` (void)
Creates a report listing all entities.

4.6.1 Detailed Description

Interface to entities functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.6.2 Function Documentation

4.6.2.1 bool db.create_entities_table (void)

Creates the entities table in the database.

Returns

true on success, false on failure.

4.6.2.2 bool db.drop_entities_table (void)

Drops the entities table in the database.

Returns

true on success, false on failure.

4.6.2.3 ds_str db.list_entities_report (void)

Creates a report listing all entities.

Returns

A [ds_str](#) containing the report.

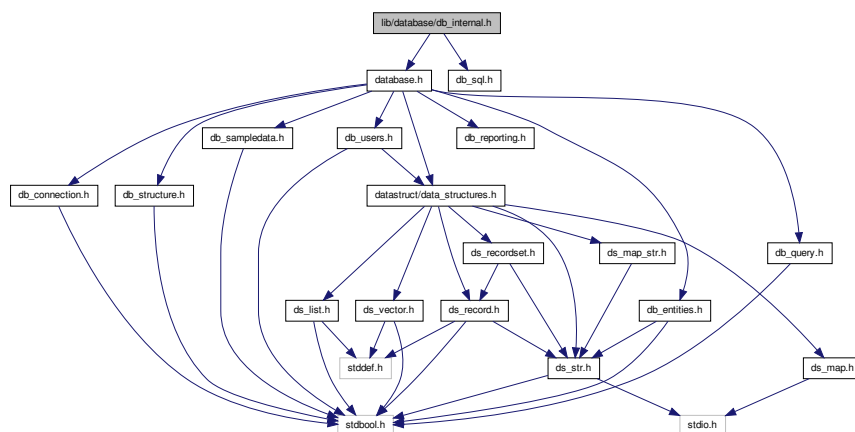
4.7 lib/database/db_internal.h File Reference

Internal library interface to database functionality.

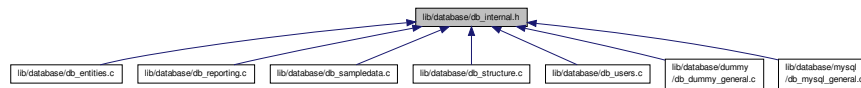
```
#include "database.h"
```

```
#include "db_sql.h"
```

Include dependency graph for db_internal.h:



This graph shows which files directly or indirectly include this file:



4.7.1 Detailed Description

Internal library interface to database functionality. The library interface includes the individual SQL functions which should be encapsulated from the user.

Author

Paul Griffiths

Copyright

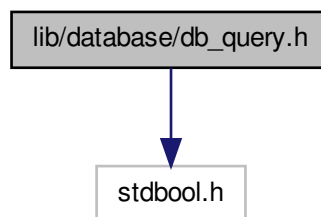
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.8 lib/database/db_query.h File Reference

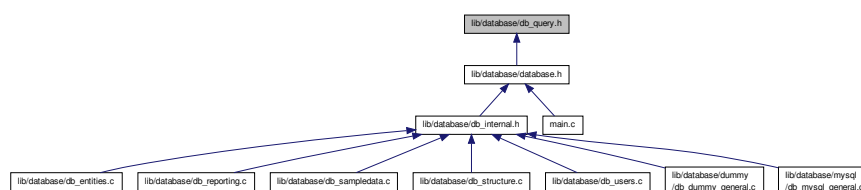
Interface to database query functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_query.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool `db_execute_query` (const char *query)
Executes an SQL query on the database.

4.8.1 Detailed Description

Interface to database query functionality. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.8.2 Function Documentation

4.8.2.1 bool db_execute_query (const char * query)

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

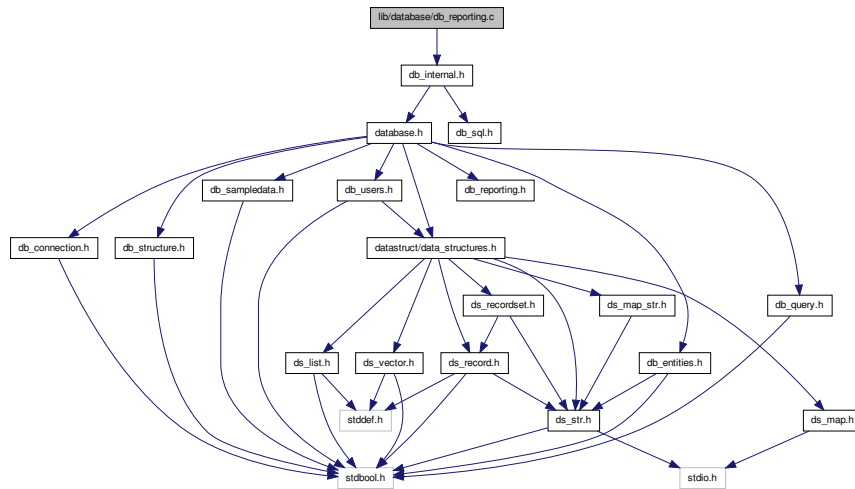
Returns

`true` if the query was successfully executed, `false` otherwise.

4.9 lib/database/db_reporting.c File Reference

Implementation of database reporting functionality.

```
#include "db_internal.h"
Include dependency graph for db_reporting.c:
```



Functions

- [ds_str db_create_report_from_query](#) (const char *query)
Creates a text report from a query.

4.9.1 Detailed Description

Implementation of database reporting functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.9.2 Function Documentation

4.9.2.1 ds_str db_create_report_from_query (const char * query)

Creates a text report from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

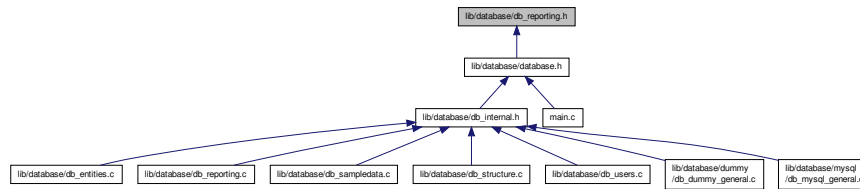
Returns

A [ds_str](#) containing the report, or NULL on failure.

4.10 lib/database/db_reporting.h File Reference

Interface to database reporting functionality.

This graph shows which files directly or indirectly include this file:



Functions

- [ds_str db_create_report_from_query](#) (const char *query)
Creates a text report from a query.
- [ds_recordset db_create_recordset_from_query](#) (const char *query)
Creates a [ds_recordset](#) from a query.

4.10.1 Detailed Description

Interface to database reporting functionality. Function implementations may be provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.10.2 Function Documentation

4.10.2.1 ds_recordset db_create_recordset_from_query (const char * query)

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or NULL on failure.

4.10.2.2 ds_str db_create_report_from_query (const char * query)

Creates a text report from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

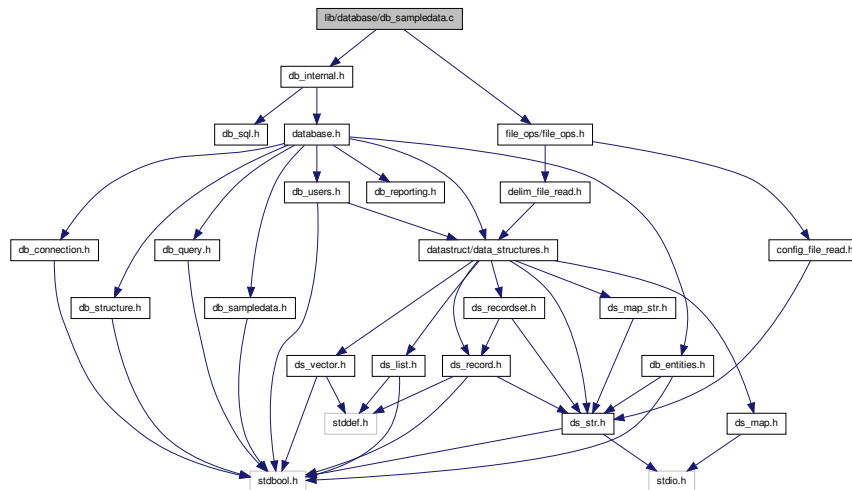
Returns

A [ds_str](#) containing the report, or NULL on failure.

4.11 lib/database/db_sampledata.c File Reference

Implementation of database sample data functionality.

```
#include "db_internal.h"
#include "file_ops/file_ops.h"
Include dependency graph for db_sampledata.c:
```



Functions

- bool [db_load_sample_data](#) (void)
Loads sample data into the database.

4.11.1 Detailed Description

Implementation of database sample data functionality.

Author

Paul Griffiths

Copyright

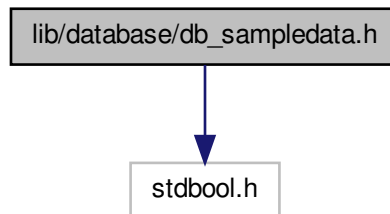
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.12 lib/database/db_sampledata.h File Reference

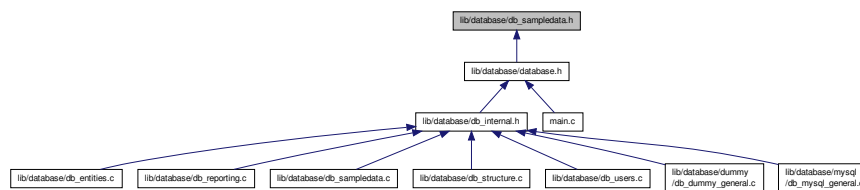
Interface to database sample data functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_sampledata.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_load_sample_data](#) (void)
Loads sample data into the database.

4.12.1 Detailed Description

Interface to database sample data functionality.

Author

Paul Griffiths

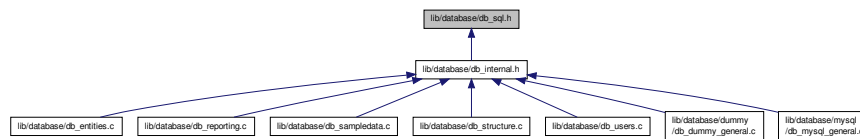
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.13 lib/database/db_sql.h File Reference

Interface to database specific SQL strings.

This graph shows which files directly or indirectly include this file:



Functions

- `const char * db_create_users_table_sql (void)`
Returns the SQL query to create the users table.
- `const char * db_drop_users_table_sql (void)`
Returns the SQL query to drop the users table.
- `const char * db_list_users_report_sql (void)`
Returns the SQL query to run the "list users" report.
- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.
- `const char * db_drop_entities_table_sql (void)`
Returns the SQL query to drop the entities table.
- `const char * db_list_entities_report_sql (void)`
Returns the SQL query to run the "list entities" report.

4.13.1 Detailed Description

Interface to database specific SQL strings. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.13.2 Function Documentation

4.13.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

4.13.2.2 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

4.13.2.3 `const char* db_drop_entities_table_sql (void)`

Returns the SQL query to drop the entities table.

Returns

The SQL query.

4.13.2.4 `const char* db_drop_users_table_sql (void)`

Returns the SQL query to drop the users table.

Returns

The SQL query.

4.13.2.5 `const char* db_list_entities_report_sql (void)`

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

4.13.2.6 `const char* db_list_users_report_sql (void)`

Returns the SQL query to run the "list users" report.

Returns

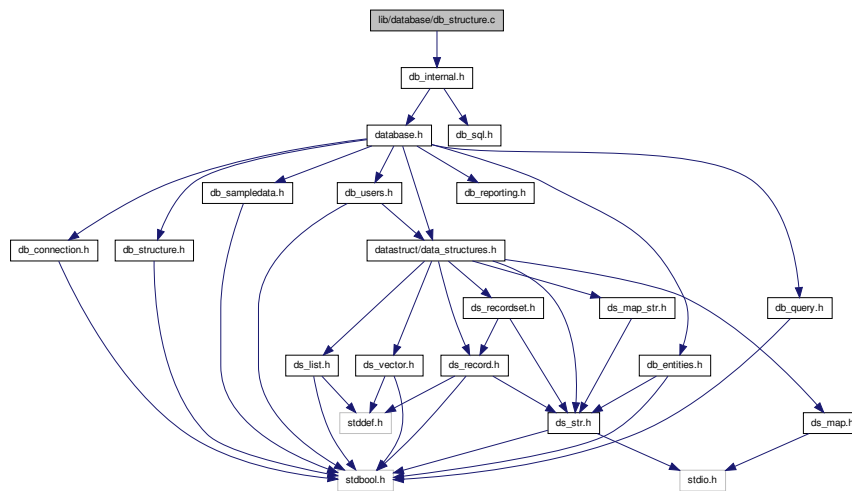
The SQL query.

4.14 lib/database/db_structure.c File Reference

Implementation of database structure functionality.

```
#include "db_internal.h"
```

Include dependency graph for db_structure.c:



Functions

- bool [db_create_database_structure](#) (void)
Creates an empty database structure.
- bool [db_delete_database_structure](#) (void)
Deletes the database structure.

4.14.1 Detailed Description

Implementation of database structure functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.14.2 Function Documentation

4.14.2.1 bool db_create_database_structure (void)

Creates an empty database structure.

Returns

true on success, false on failure.

4.14.2.2 bool db_delete_database_structure (void)

Deletes the database structure.

Returns

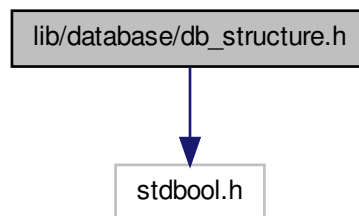
true on success, false on failure.

4.15 lib/database/db_structure.h File Reference

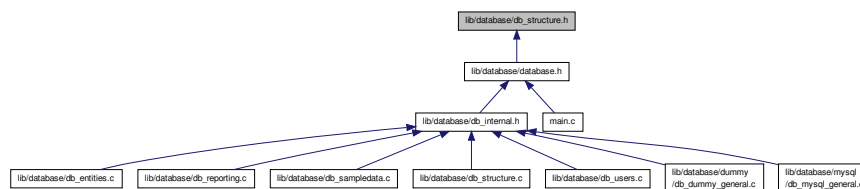
Interface to database structure functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_structure.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_create_database_structure](#) (void)
Creates an empty database structure.
- bool [db_delete_database_structure](#) (void)
Deletes the database structure.

4.15.1 Detailed Description

Interface to database structure functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.15.2 Function Documentation**4.15.2.1 bool db_create_database_structure (void)**

Creates an empty database structure.

Returns

true on success, false on failure.

4.15.2.2 bool db_delete_database_structure (void)

Deletes the database structure.

Returns

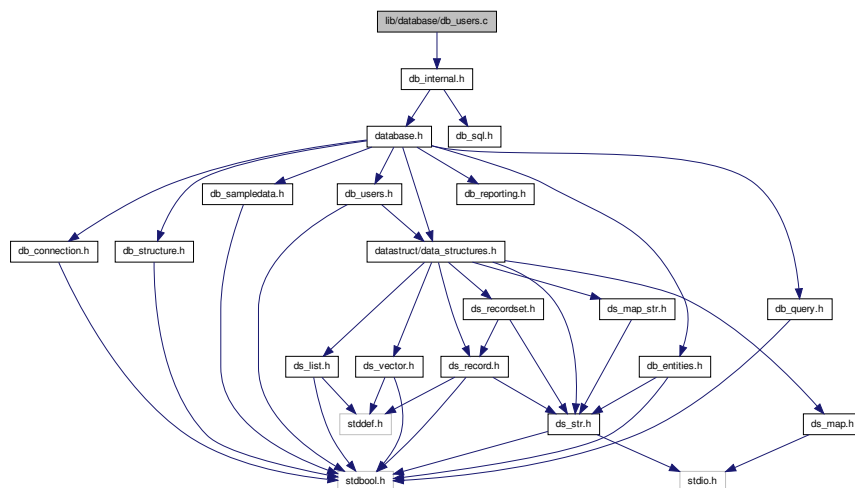
true on success, false on failure.

4.16 lib/database/db_users.c File Reference

Implementation of users functionality.

```
#include "db_internal.h"
```

Include dependency graph for db_users.c:



Functions

- `bool db_create_users_table (void)`
Creates the users table in the database.
- `bool db_drop_users_table (void)`
Drops the users table from the database.
- `ds_str db_list_users_report (void)`
Creates a report listing all users.

4.16.1 Detailed Description

Implementation of users functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.16.2 Function Documentation

4.16.2.1 `bool db_create_users_table (void)`

Creates the users table in the database.

Returns

`true` on success, `false` on failure.

4.16.2.2 `bool db_drop_users_table (void)`

Drops the users table from the database.

Returns

`true` on success, `false` on failure.

4.16.2.3 `ds_str db_list_users_report (void)`

Creates a report listing all users.

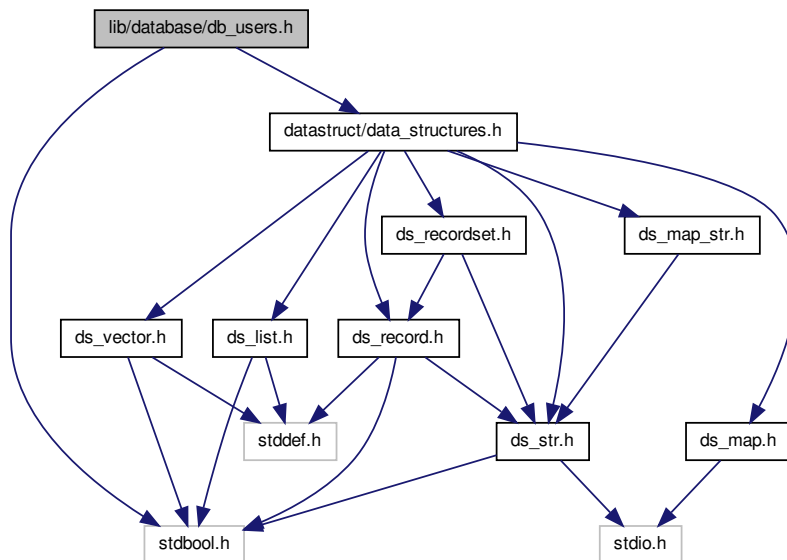
Returns

A `ds_str` containing the report.

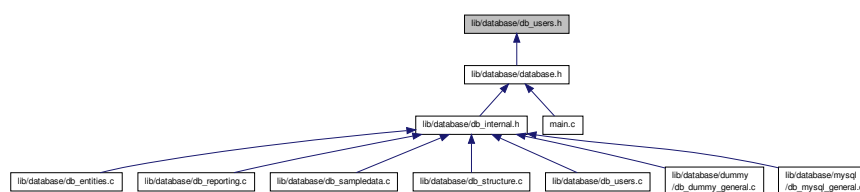
4.17 lib/database/db_users.h File Reference

Interface to users functionality.

```
#include <stdbool.h>
#include "datastruct/data_structures.h"
Include dependency graph for db_users.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_create_users_table](#) (void)
Creates the users table in the database.
- bool [db_drop_users_table](#) (void)
Drops the users table from the database.
- [ds_str db_list_users_report](#) (void)
Creates a report listing all users.

4.17.1 Detailed Description

Interface to users functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.17.2 Function Documentation**4.17.2.1 bool db_create_users_table (void)**

Creates the users table in the database.

Returns

`true` on success, `false` on failure.

4.17.2.2 bool db_drop_users_table (void)

Drops the users table from the database.

Returns

`true` on success, `false` on failure.

4.17.2.3 ds_str db_list_users_report (void)

Creates a report listing all users.

Returns

A `ds_str` containing the report.

4.18 lib/database/dummy/db_dummy_create_entities_table_sql.c File Reference

Returns dummy SQL query to create entities table.

Functions

- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.

4.18.1 Detailed Description

Returns dummy SQL query to create entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.18.2 Function Documentation

4.18.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

4.19 `lib/database/dummy/db_dummy_create_users_table_sql.c` File Reference

Returns dummy SQL query to create users table.

Functions

- `const char * db_create_users_table_sql (void)`
Returns the SQL query to create the users table.

4.19.1 Detailed Description

Returns dummy SQL query to create users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.19.2 Function Documentation

4.19.2.1 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

4.20 `lib/database/dummy/db_dummy_drop_entities_table_sql.c` File Reference

Returns dummy SQL query to drop entities table.

Functions

- `const char * db_drop_entities_table_sql (void)`
Returns the SQL query to drop the entities table.

4.20.1 Detailed Description

Returns dummy SQL query to drop entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.20.2 Function Documentation

4.20.2.1 `const char* db_drop_entities_table_sql (void)`

Returns the SQL query to drop the entities table.

Returns

The SQL query.

4.21 lib/database/dummy/db_dummy_drop_users_table_sql.c File Reference

Returns dummy SQL query to drop users table.

Functions

- `const char * db_drop_users_table_sql (void)`
Returns the SQL query to drop the users table.

4.21.1 Detailed Description

Returns dummy SQL query to drop users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.21.2 Function Documentation

4.21.2.1 `const char* db_drop_users_table_sql (void)`

Returns the SQL query to drop the users table.

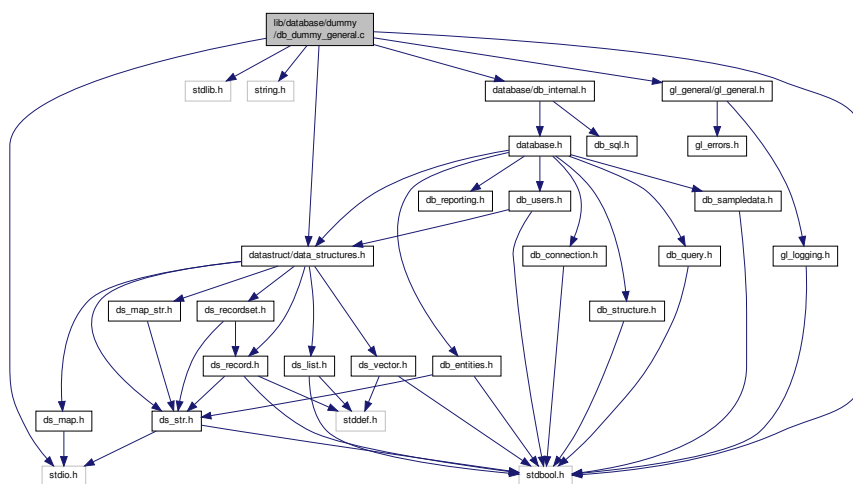
Returns

The SQL query.

4.22 lib/database/dummy/db_dummy_general.c File Reference

Implementation of dummy database functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
#include "datastruct/data_structures.h"
Include dependency graph for db_dummy_general.c:
```



Macros

- `#define _XOPEN_SOURCE 600`

Functions

- `bool db_connect (const char *host, const char *database, const char *username, const char *password)`
Connects to a database.
- `void db_close (void)`
Disconnects from a database.
- `bool db_execute_query (const char *query)`
Executes an SQL query on the database.
- `ds_recordset db_create_recordset_from_query (const char *query)`
Creates a [ds_recordset](#) from a query.

4.22.1 Detailed Description

Implementation of dummy database functionality. This module is useful when compiling for testing purpose on a system without any of the supported database development libraries available.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.22.2 Macro Definition Documentation

4.22.2.1 #define _XOPEN_SOURCE 600

UNIX feature test macro

4.22.3 Function Documentation

4.22.3.1 bool db_connect (const char * *host*, const char * *database*, const char * *username*, const char * *password*)

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

`true` if the connection was successfully made, `false` otherwise.

4.22.3.2 ds_recordset db_create_recordset_from_query (const char * *query*)

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or `NULL` on failure.

4.22.3.3 bool db_execute_query (const char * *query*)

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

Returns

`true` if the query was successfully executed, `false` otherwise.

4.23 lib/database/dummy/db_dummy_list_entities_report_sql.c File Reference

Returns dummy SQL query to create list entities report.

Functions

- `const char * db_list_entities_report_sql (void)`
Returns the SQL query to run the "list entities" report.

4.23.1 Detailed Description

Returns dummy SQL query to create list entities report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.23.2 Function Documentation

4.23.2.1 `const char* db_list_entities_report_sql (void)`

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

4.24 lib/database/dummy/db_dummy_list_users_report_sql.c File Reference

Returns dummy SQL query to create list users report.

Functions

- `const char * db_list_users_report_sql (void)`
Returns the SQL query to run the "list users" report.

4.24.1 Detailed Description

Returns dummy SQL query to create list users report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.24.2 Function Documentation

4.24.2.1 `const char* db_list_users_report_sql (void)`

Returns the SQL query to run the "list users" report.

Returns

The SQL query.

4.25 lib/database/mysql/db_mysql_create_entities_table_sql.c File Reference

Returns MYSQL SQL query to create entities table.

Functions

- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.

4.25.1 Detailed Description

Returns MYSQL SQL query to create entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.25.2 Function Documentation

4.25.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

4.26 lib/database/mysql/db_mysql_create_users_table_sql.c File Reference

Returns MYSQL SQL query to create users table.

Functions

- `const char * db_create_users_table_sql` (void)
Returns the SQL query to create the users table.

4.26.1 Detailed Description

Returns MYSQL SQL query to create users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.26.2 Function Documentation

4.26.2.1 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

4.27 lib/database/mysql/db_mysql_drop_entities_table_sql.c File Reference

Returns MYSQL SQL query to drop entities table.

Functions

- `const char * db_drop_entities_table_sql` (void)
Returns the SQL query to drop the entities table.

4.27.1 Detailed Description

Returns MYSQL SQL query to drop entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.27.2 Function Documentation

4.27.2.1 const char* db_drop_entities_table_sql (void)

Returns the SQL query to drop the entities table.

Returns

The SQL query.

4.28 lib/database/mysql/db_mysql_drop_users_table_sql.c File Reference

Returns MYSQL SQL query to drop users table.

Functions

- const char * [db_drop_users_table_sql](#) (void)
Returns the SQL query to drop the users table.

4.28.1 Detailed Description

Returns MYSQL SQL query to drop users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.28.2 Function Documentation

4.28.2.1 const char* db_drop_users_table_sql (void)

Returns the SQL query to drop the users table.

Returns

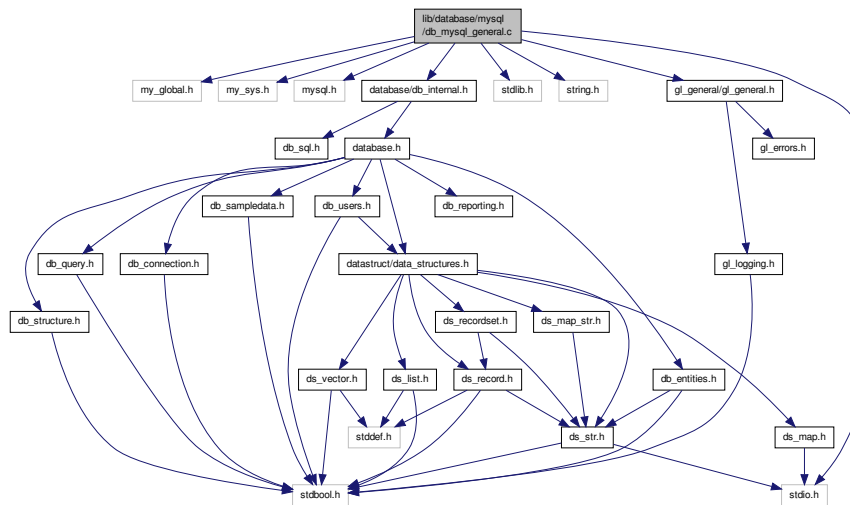
The SQL query.

4.29 lib/database/mysql/db_mysql_general.c File Reference

Implementation of MYSQL database functionality.

```
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
```

Include dependency graph for `db_mysql_general.c`:



Functions

- bool `db_connect` (const char *host, const char *database, const char *username, const char *password)
Connects to a database.
- void `db_close` (void)
Disconnects from a database.
- bool `db_execute_query` (const char *query)
Executes an SQL query on the database.
- `ds_recordset db_create_recordset_from_query` (const char *query)
Creates a `ds_recordset` from a query.

Variables

- MYSQL * `main_mss` = NULL
- MYSQL * `conn_mss` = NULL

4.29.1 Detailed Description

Implementation of MYSQL database functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.29.2 Function Documentation

4.29.2.1 `bool db_connect (const char * host, const char * database, const char * username, const char * password)`

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

`true` if the connection was successfully made, `false` otherwise.

4.29.2.2 `ds_recordset db_create_recordset_from_query (const char * query)`

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or `NULL` on failure.

4.29.2.3 `bool db_execute_query (const char * query)`

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

Returns

`true` if the query was successfully executed, `false` otherwise.

4.29.3 Variable Documentation

4.29.3.1 `MYSQL* conn_mss = NULL`

MYSQL connection object.

4.29.3.2 `MYSQL* main_mss = NULL`

MYSQL initialization object.

4.30 lib/database/mysql/db_mysql_list_entities_report_sql.c File Reference

Returns MYSQL SQL query to create list entities report.

Functions

- const char * [db_list_entities_report_sql](#) (void)
Returns the SQL query to run the "list entities" report.

4.30.1 Detailed Description

Returns MYSQL SQL query to create list entities report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.30.2 Function Documentation

4.30.2.1 const char* db_list_entities_report_sql (void)

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

4.31 lib/database/mysql/db_mysql_list_users_report_sql.c File Reference

Returns MYSQL SQL query to create list users report.

Functions

- const char * [db_list_users_report_sql](#) (void)
Returns the SQL query to run the "list users" report.

4.31.1 Detailed Description

Returns MYSQL SQL query to create list users report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.32.1 Detailed Description

Interface to data structures.

Author

Paul Griffiths

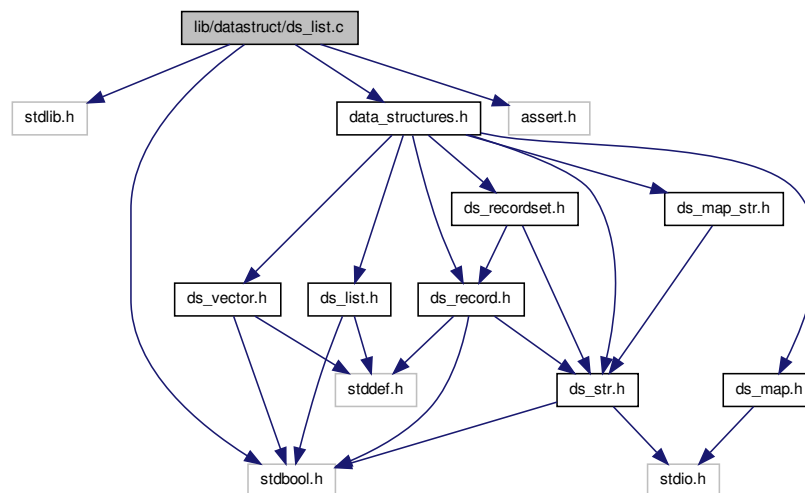
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.33 lib/datastruct/ds_list.c File Reference

Implementation of generic doubly-linked list data structure.

```
#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_list.c:
```



Data Structures

- struct [ds_list_element](#)
- struct [ds_list](#)

Functions

- [ds_list ds_list_create](#) (const bool free_on_delete, void(*destructor)(void *))
Creates a new list.
- void [ds_list_destroy](#) ([ds_list](#) list)
Destroys a list and frees any associated resources.

- void `ds_list_destructor` (void *list)
A list destructor function.
- `ds_list ds_list_append` (`ds_list` list, void *data)
Appends an element to a list.
- void `ds_list_remove_tail` (`ds_list` list)
Removes the last element of a list.
- void `ds_list_remove_all` (`ds_list` list)
Removes all the elements from a list.
- void * `ds_list_element` (`ds_list` list, const size_t index)
Retrieves the data at a specified index.
- size_t `ds_list_length` (`ds_list` list)
Returns the number of elements in a list.
- bool `ds_list_is_empty` (`ds_list` list)
Checks if a list is empty.
- void `ds_list_seek_start` (`ds_list` list)
Sets the current element to the first element of a list.
- void `ds_list_seek_end` (`ds_list` list)
Sets the current element to the last element of a list.
- void * `ds_list_get_next_data` (`ds_list` list)
Returns the next element of the list.
- void * `ds_list_get_prev_data` (`ds_list` list)
Returns the previous element of the list.

4.33.1 Detailed Description

Implementation of generic doubly-linked list data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.33.2 Function Documentation

4.33.2.1 `ds_list ds_list_append (ds_list list, void * element)`

Appends an element to a list.

Parameters

<i>list</i>	The list to which to append.
<i>element</i>	The element to append.

Returns

The same list, or NULL on failure.

4.33.2.2 `ds_list ds_list_create (const bool free_on_delete, void(*)(void *) destructor)`

Creates a new list.

Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created list, or `NULL` on failure.

4.33.2.3 `void ds_list_destroy (ds_list list)`

Destroys a list and frees any associated resources.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

4.33.2.4 `void ds_list_destructor (void * list)`

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

4.33.2.5 `void* ds_list_element (ds_list list, const size_t index)`

Retrieves the data at a specified index.

Parameters

<i>list</i>	The list from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

4.33.2.6 `void* ds_list_get_next_data (ds_list list)`

Returns the next element of the list.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the next element, or `NULL` if the end of the list has been reached.

4.33.2.7 void* ds_list_get_prev_data (ds_list list)

Returns the previous element of the list.

This function returns the data of the "current element", and decrements the current element pointer. Subsequent calls to this function will return successively earlier elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the previous element, or `NULL` if the start of the list has been reached.

4.33.2.8 bool ds_list_is_empty (ds_list list)

Checks if a list is empty.

Parameters

<i>list</i>	The list to check.
-------------	--------------------

Returns

`true` if the list is empty, `false` otherwise.

4.33.2.9 size_t ds_list_length (ds_list list)

Returns the number of elements in a list.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

The number of elements in the list.

4.33.2.10 void ds_list_remove_all (ds_list list)

Removes all the elements from a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.33.2.11 void ds_list_remove_tail (ds_list list)

Removes the last element of a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.33.2.12 void ds_list_seek_end (ds_list list)

Sets the current element to the last element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

4.33.2.13 void ds_list_seek_start (ds_list list)

Sets the current element to the first element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

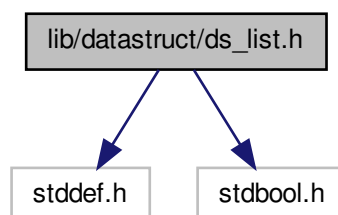
4.34 lib/datastruct/ds_list.h File Reference

Interface to generic doubly-linked list data structure.

```
#include <stddef.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_list.h:



[illegible]

- typedef struct ds_list * ds_list

- `ds_list ds_list_create` (const bool free_on_delete, void(*destructor)(void *))
Creates a new list.
- void `ds_list_destroy` (`ds_list` list)
Destroys a list and frees any associated resources.
- void `ds_list_destructor` (void *list)
A list destructor function.
- `ds_list ds_list_append` (`ds_list` list, void *element)
Appends an element to a list.
- void `ds_list_remove_tail` (`ds_list` list)
Removes the last element of a list.
- void `ds_list_remove_all` (`ds_list` list)
Removes all the elements from a list.
- void * `ds_list_element` (`ds_list` list, const size_t index)
Retrieves the data at a specified index.
- size_t `ds_list_length` (`ds_list` list)
Returns the number of elements in a list.
- bool `ds_list_is_empty` (`ds_list` list)
Checks if a list is empty.
- void `ds_list_seek_start` (`ds_list` list)
Sets the current element to the first element of a list.
- void `ds_list_seek_end` (`ds_list` list)
Sets the current element to the last element of a list.
- void * `ds_list_get_next_data` (`ds_list` list)
Returns the next element of the list.
- void * `ds_list_get_prev_data` (`ds_list` list)
Returns the previous element of the list.

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.34.2 Typedef Documentation

4.34.2.1 typedef struct ds_list* ds_list

Typedef for opaque list datatype

4.34.3 Function Documentation

4.34.3.1 ds_list ds_list_append (ds_list *list*, void * *element*)

Appends an element to a list.

Parameters

<i>list</i>	The list to which to append.
<i>element</i>	The element to append.

Returns

The same list, or `NULL` on failure.

4.34.3.2 ds_list ds_list_create (const bool *free_on_delete*, void(*) (void *) *destructor*)

Creates a new list.

Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created list, or `NULL` on failure.

4.34.3.3 void ds_list_destroy (ds_list *list*)

Destroys a list and frees any associated resources.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

4.34.3.4 void ds_list_destructor (void * *list*)

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

4.34.3.5 void* ds_list_element (ds_list *list*, const size_t *index*)

Retrieves the data at a specified index.

Parameters

<i>list</i>	The list from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

4.34.3.6 void* ds_list_get_next_data (ds_list *list*)

Returns the next element of the list.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the next element, or `NULL` if the end of the list has been reached.

4.34.3.7 void* ds_list_get_prev_data (ds_list *list*)

Returns the previous element of the list.

This function returns the data of the "current element", and decrements the current element pointer. Subsequent calls to this function will return successively earlier elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the previous element, or `NULL` if the start of the list has been reached.

4.34.3.8 `bool ds_list_is_empty (ds_list list)`

Checks if a list is empty.

Parameters

<i>list</i>	The list to check.
-------------	--------------------

Returns

`true` is the list is empty, `false` otherwise.

4.34.3.9 `size_t ds_list_length (ds_list list)`

Returns the number of elements in a list.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

The number of elements in the list.

4.34.3.10 `void ds_list_remove_all (ds_list list)`

Removes all the elements from a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.34.3.11 `void ds_list_remove_tail (ds_list list)`

Removes the last element of a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

4.34.3.12 `void ds_list_seek_end (ds_list list)`

Sets the current element to the last element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

4.34.3.13 `void ds_list_seek_start (ds_list list)`

Sets the current element to the first element of a list.

Parameters

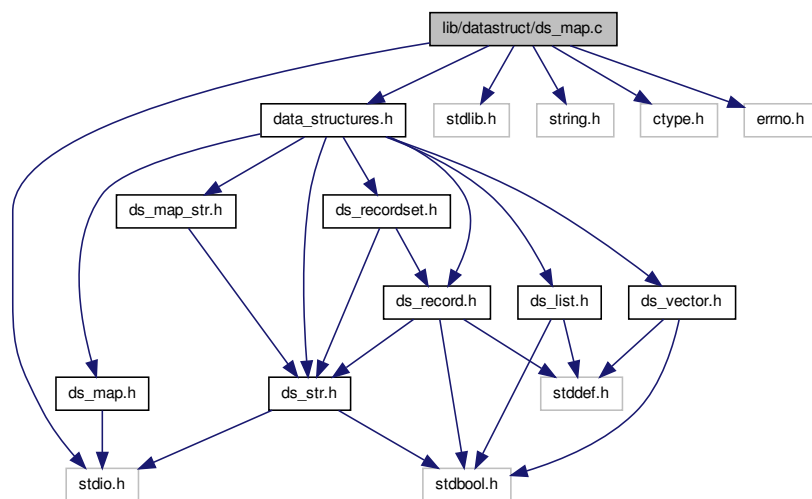
<i>list</i>	The list.
-------------	-----------

4.35 lib/datastruct/ds_map.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include "data_structures.h"
```

Include dependency graph for ds_map.c:



Data Structures

- struct [kv_pair_node](#)
- struct [ds_map](#)

Macros

- `#define _POSIX_C_SOURCE 200809L`
Enables POSIX library functions.

Functions

- `ds_map ds_map_init (const size_t hash_size)`
Initializes a hash map.
- `void ds_map_destroy (ds_map map)`
Destroys a hash map.
- `const char * ds_map_get_value (ds_map map, const char *key)`

Retrieves a value associated with a key in the map.

- void `ds_map_insert` (`ds_map` map, const char *key, const char *value)

Inserts a key-value pair into a map.

- void `ds_map_print_all` (`ds_map` map, FILE *outfile)

Prints all the key-value pairs in a map to stdout.

4.35.1 Detailed Description

Implementation of string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.35.2 Function Documentation

4.35.2.1 void `ds_map_destroy` (`ds_map` map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

4.35.2.2 const char* `ds_map_get_value` (`ds_map` map, const char * key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.35.2.3 `ds_map` `ds_map_init` (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

4.35.2.4 void ds_map_insert (ds_map map, const char * key, const char * value)

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

4.35.2.5 void ds_map_print_all (ds_map map, FILE * outfile)

Prints all the key-value pairs in a map to stdout.

Parameters

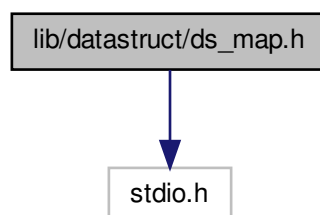
<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

4.36 lib/datastruct/ds_map.h File Reference

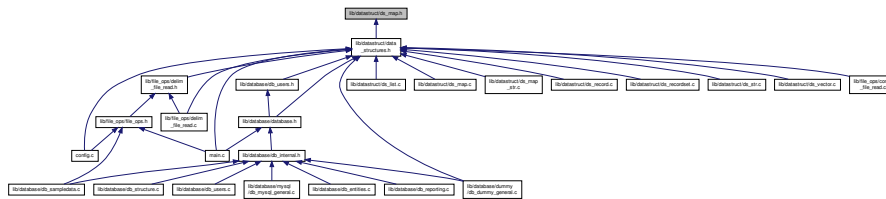
Interface to string-string hash map data structure.

```
#include <stdio.h>
```

Include dependency graph for ds_map.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_map](#) * [ds_map](#)

Functions

- [ds_map ds_map_init](#) (const size_t hash_size)
Initializes a hash map.
- void [ds_map_destroy](#) ([ds_map](#) map)
Destroys a hash map.
- const char * [ds_map_get_value](#) ([ds_map](#) map, const char *key)
Retrieves a value associated with a key in the map.
- void [ds_map_insert](#) ([ds_map](#) map, const char *key, const char *value)
Inserts a key-value pair into a map.
- void [ds_map_print_all](#) ([ds_map](#) map, FILE *outfile)
Prints all the key-value pairs in a map to stdout.

4.36.1 Detailed Description

Interface to string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.36.2 Typedef Documentation

4.36.2.1 typedef struct [ds_map](#)* [ds_map](#)

Opaque data type for hash map

4.36.3 Function Documentation

4.36.3.1 void [ds_map_destroy](#) ([ds_map](#) map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

4.36.3.2 `const char* ds_map_get_value (ds_map map, const char * key)`

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.36.3.3 `ds_map ds_map_init (const size_t hash_size)`

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

4.36.3.4 `void ds_map_insert (ds_map map, const char * key, const char * value)`

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

4.36.3.5 `void ds_map_print_all (ds_map map, FILE * outfile)`

Prints all the key-value pairs in a map to stdout.

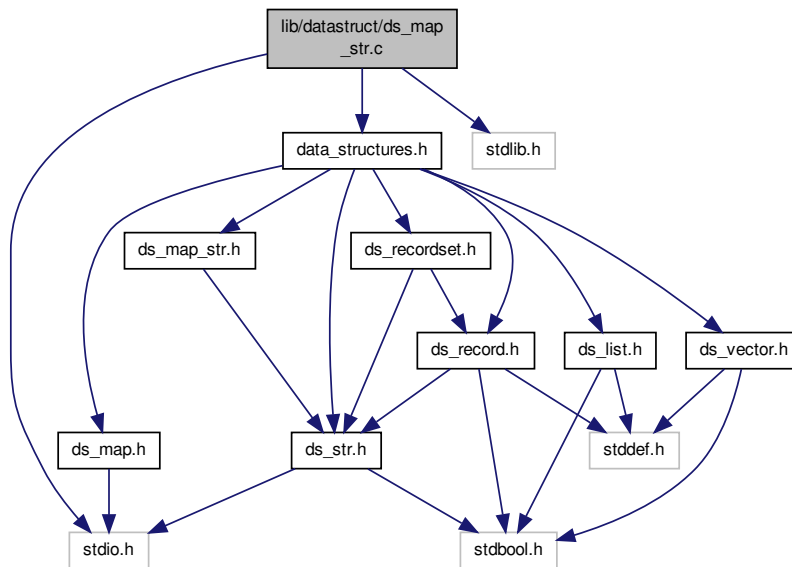
Parameters

<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

4.37 lib/datastruct/ds_map_str.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include "data_structures.h"
Include dependency graph for ds_map_str.c:
```



Data Structures

- struct [kv_pair_node](#)
- struct [ds_map_str](#)

Functions

- [ds_map_str ds_map_str_init](#) (const [size_t](#) hash_size)
Initializes a hash map.
- void [ds_map_str_destroy](#) ([ds_map_str](#) map)
Destroys a hash map.
- [ds_str ds_map_str_get_value](#) ([ds_map_str](#) map, [ds_str](#) key)
Retrieves a value associated with a key in the map.
- void [ds_map_str_insert](#) ([ds_map_str](#) map, [ds_str](#) key, [ds_str](#) value)
Inserts a key-value pair into a map.

4.37.1 Detailed Description

Implementation of string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.37.2 Function Documentation

4.37.2.1 void ds_map_str_destroy (ds_map_str map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

4.37.2.2 ds_str ds_map_str_get_value (ds_map_str map, ds_str key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.37.2.3 ds_map_str ds_map_str_init (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

4.37.2.4 void ds_map_str_insert (ds_map_str map, ds_str key, ds_str value)

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

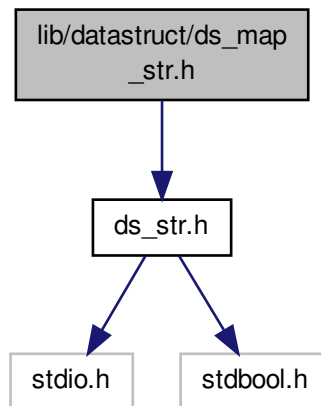
<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

4.38 lib/datastruct/ds_map_str.h File Reference

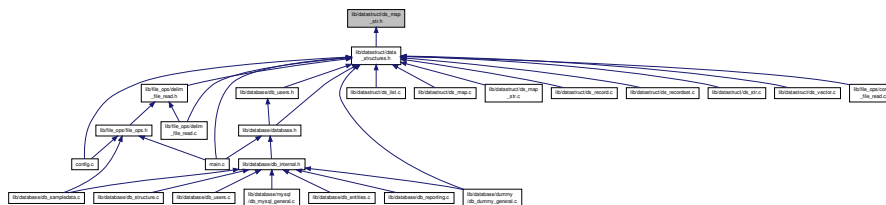
Interface to string-string hash map data structure.

```
#include "ds_str.h"
```

Include dependency graph for ds_map_str.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_map_str](#) * [ds_map_str](#)

Functions

- [ds_map_str](#) [ds_map_str_init](#) (const [size_t](#) hash_size)
Initializes a hash map.
- void [ds_map_str_destroy](#) ([ds_map_str](#) map)
Destroys a hash map.
- [ds_str](#) [ds_map_str_get_value](#) ([ds_map_str](#) map, [ds_str](#) key)
Retrieves a value associated with a key in the map.
- void [ds_map_str_insert](#) ([ds_map_str](#) map, [ds_str](#) key, [ds_str](#) value)
Inserts a key-value pair into a map.

4.38.1 Detailed Description

Interface to string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.38.2 Typedef Documentation

4.38.2.1 typedef struct ds_map_str* ds_map_str

Opaque data type for hash map

4.38.3 Function Documentation

4.38.3.1 void ds_map_str_destroy (ds_map_str map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

4.38.3.2 ds_str ds_map_str_get_value (ds_map_str map, ds_str key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

4.38.3.3 ds_map_str ds_map_str_init (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

4.38.3.4 void ds_map_str_insert (ds_map_str map, ds_str key, ds_str value)

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

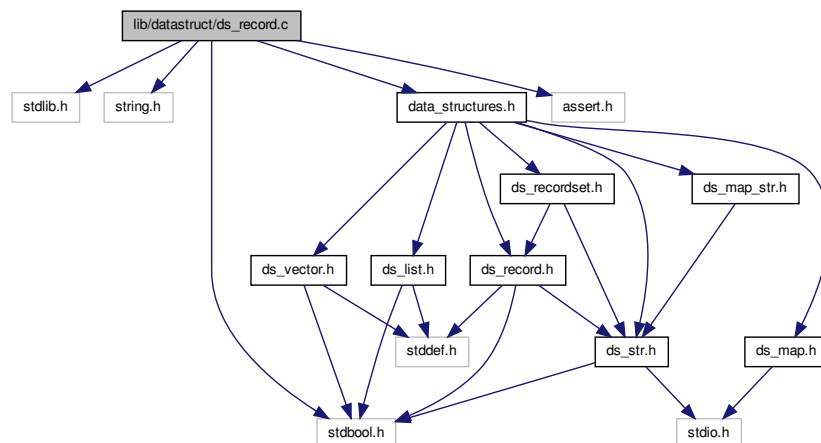
Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

4.39 lib/datastruct/ds_record.c File Reference

Implementation of record database structure.

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_record.c:
```



Data Structures

- struct [ds_record](#)

Functions

- [ds_record ds_record_create](#) (const size_t size)
Creates a new record.
- void [ds_record_destroy](#) (ds_record record)

- Destroys a record and frees any associated resources.*
- void `ds_record_destructor` (void *record)
A record destructor function.
- void `ds_record_clear` (ds_record record)
Clears and `free()`s all the elements in a record.
- void `ds_record_set_field` (ds_record record, const size_t index, ds_str field)
Sets a field of a record.
- ds_str `ds_record_get_field` (ds_record record, const size_t index)
Retrieves the field at a specified index.
- size_t `ds_record_size` (ds_record record)
Returns the size of a record.
- void `ds_record_seek_start` (ds_record record)
Sets the current field to the first field of a record.
- ds_str `ds_record_get_next_data` (ds_record record)
Returns the next field of the record.
- ds_record `ds_record_tokenize` (ds_str str, const char delim)
Tokenizes a string into a record.
- ds_str `ds_record_make_delim_string` (ds_record record, const char delim)
Makes a delimited string from a record.
- ds_str `ds_record_make_values_string` (ds_record record)
Makes a delimited SQL values string from a record.

4.39.1 Detailed Description

Implementation of record database structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.39.2 Function Documentation

4.39.2.1 void ds_record_clear (ds_record record)

Clears and `free()`s all the elements in a record.

Parameters

<i>record</i>	The record.
---------------	-------------

4.39.2.2 ds_record ds_record_create (const size_t size)

Creates a new record.

Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

Returns

A newly created record, or `NULL` on failure.

4.39.2.3 void ds_record_destroy (ds_record record)

Destroys a record and frees any associated resources.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

4.39.2.4 void ds_record_destructor (void * record)

A record destructor function.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

4.39.2.5 ds_str ds_record_get_field (ds_record record, const size_t index)

Retrieves the field at a specified index.

Parameters

<i>record</i>	The record from which to retrieve.
<i>index</i>	The index of the desired field.

Returns

A pointer to the field, or `NULL` if the index is out of range.

4.39.2.6 ds_str ds_record_get_next_data (ds_record record)

Returns the next field of the record.

This function returns the data of the "current field", and advances the current field pointer. Subsequent calls to this function will return successive fields.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

A pointer to the next field, or `NULL` if the end of the record has been reached.

4.39.2.7 ds_str ds_record_make_delim_string (ds_record record, const char delim)

Makes a delimited string from a record.

Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

Returns

The delimited string, or `NULL` on failure.

4.39.2.8 ds_str ds_record_make_values_string (ds_record record)

Makes a delimited SQL values string from a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The delimited values string, or `NULL` on failure.

4.39.2.9 void ds_record_seek_start (ds_record record)

Sets the current field to the first field of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

4.39.2.10 void ds_record_set_field (ds_record record, const size_t index, ds_str field)

Sets a field of a record.

If the field is currently occupied, the existing field is `free()`d.

Parameters

<i>record</i>	The record to set.
<i>index</i>	The index of the field to set.
<i>field</i>	The value to which to set the field.

4.39.2.11 size_t ds_record_size (ds_record record)

Returns the size of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The size of the record.

Functions

- [ds_record ds_record_create](#) (const size_t size)
Creates a new record.
- void [ds_record_destroy](#) ([ds_record](#) record)
Destroys a record and frees any associated resources.
- void [ds_record_destructor](#) (void *record)
A record destructor function.
- void [ds_record_clear](#) ([ds_record](#) record)
Clears and `free()`s all the elements in a record.
- void [ds_record_set_field](#) ([ds_record](#) record, const size_t index, [ds_str](#) field)
Sets a field of a record.
- [ds_str ds_record_get_field](#) ([ds_record](#) record, const size_t index)
Retrieves the field at a specified index.
- size_t [ds_record_size](#) ([ds_record](#) record)
Returns the size of a record.
- void [ds_record_seek_start](#) ([ds_record](#) record)
Sets the current field to the first field of a record.
- [ds_str ds_record_get_next_data](#) ([ds_record](#) record)
Returns the next field of the record.
- [ds_record ds_record_tokenize](#) ([ds_str](#) str, const char delim)
Tokenizes a string into a record.
- [ds_str ds_record_make_delim_string](#) ([ds_record](#) record, const char delim)
Makes a delimited string from a record.
- [ds_str ds_record_make_values_string](#) ([ds_record](#) record)
Makes a delimited SQL values string from a record.

4.40.1 Detailed Description

Interface to record data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.40.2 Typedef Documentation

4.40.2.1 typedef struct ds_record* ds_record

Typedef for opaque record datatype

4.40.3 Function Documentation

4.40.3.1 void ds_record_clear (ds_record record)

Clears and `free()`s all the elements in a record.

Parameters

<i>record</i>	The record.
---------------	-------------

4.40.3.2 `ds_record ds_record_create (const size_t size)`

Creates a new record.

Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

Returns

A newly created record, or `NULL` on failure.

4.40.3.3 `void ds_record_destroy (ds_record record)`

Destroys a record and frees any associated resources.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

4.40.3.4 `void ds_record_destructor (void * record)`

A record destructor function.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

4.40.3.5 `ds_str ds_record_get_field (ds_record record, const size_t index)`

Retrieves the field at a specified index.

Parameters

<i>record</i>	The record from which to retrieve.
<i>index</i>	The index of the desired field.

Returns

A pointer to the field, or `NULL` if the index is out of range.

4.40.3.6 `ds_str ds_record_get_next_data (ds_record record)`

Returns the next field of the record.

This function returns the data of the "current field", and advances the current field pointer. Subsequent calls to this function will return successive fields.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

A pointer to the next field, or `NULL` if the end of the record has been reached.

4.40.3.7 ds_str ds_record_make_delim_string (ds_record record, const char delim)

Makes a delimited string from a record.

Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

Returns

The delimited string, or `NULL` on failure.

4.40.3.8 ds_str ds_record_make_values_string (ds_record record)

Makes a delimited SQL values string from a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The delimited values string, or `NULL` on failure.

4.40.3.9 void ds_record_seek_start (ds_record record)

Sets the current field to the first field of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

4.40.3.10 void ds_record_set_field (ds_record record, const size_t index, ds_str field)

Sets a field of a record.

If the field is currently occupied, the existing field is `free()`d.

Parameters

<i>record</i>	The record to set.
<i>index</i>	The index of the field to set.
<i>field</i>	The value to which to set the field.

4.40.3.11 `size_t ds_record_size (ds_record record)`

Returns the size of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The size of the record.

4.40.3.12 `ds_record ds_record_tokenize (ds_str str, const char delim)`

Tokenizes a string into a record.

Parameters

<i>str</i>	The string to tokenize.
<i>delim</i>	The delimiting character.

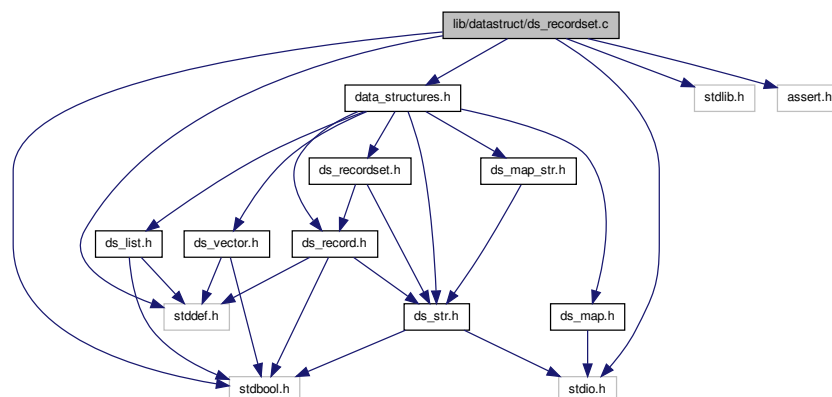
Returns

A new record containing the tokens.

4.41 lib/datastruct/ds_recordset.c File Reference

Implementation of query result set structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_recordset.c:
```



Data Structures

- struct [ds_recordset](#)

Functions

- [ds_recordset ds_recordset_create](#) (const size_t num_fields)
Creates a new record set.
- void [ds_recordset_destroy](#) (ds_recordset set)
Destroys a record set and frees associated resources.
- [ds_record ds_recordset_add_record](#) (ds_recordset set, ds_record record)
Adds a record to a record set.
- size_t [ds_recordset_num_fields](#) (ds_recordset set)
Returns the number of fields in a record set.
- size_t [ds_recordset_num_records](#) (ds_recordset set)
Returns the number of records in a record set.
- void [ds_recordset_set_headers](#) (ds_recordset set, ds_record headers)
Sets the record headers in a record set.
- [ds_str ds_recordset_get_text_report](#) (ds_recordset set)
Returns a formatted text report for the record set.
- void [ds_recordset_seek_start](#) (ds_recordset set)
Sets the current record to the first record.
- [ds_record ds_recordset_next_record](#) (ds_recordset set)
Returns the next record in the record set.
- [ds_str ds_recordset_get_next_insert_query](#) (ds_recordset set, const char *table_name)
Gets the next SQL INSERT query.

4.41.1 Detailed Description

Implementation of query result set structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.41.2 Function Documentation

4.41.2.1 ds_record ds_recordset_add_record (ds_recordset set, ds_record record)

Adds a record to a record set.

The record *must* have the same number of fields as the number of fields provided to [ds_recordset_create\(\)](#).

Parameters

<i>set</i>	The record set to which to add.
<i>record</i>	The record to add.

Returns

A pointer to the new record (i.e. it returns the second parameter) or `NULL` on failure.

4.41.2.2 `ds_recordset ds_recordset_create (const size_t num_fields)`

Creates a new record set.

Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

Returns

A pointer to the new record set.

4.41.2.3 `void ds_recordset_destroy (ds_recordset set)`

Destroys a record set and frees associated resources.

Parameters

<i>set</i>	The record set to destroy.
------------	----------------------------

4.41.2.4 `ds_str ds_recordset_get_next_insert_query (ds_recordset set, const char * table_name)`

Gets the next SQL INSERT query.

Parameters

<i>set</i>	The set.
<i>table_name</i>	The table name into which to insert.

Returns

The query. Caller is responsible for `free()` ing.

4.41.2.5 `ds_str ds_recordset_get_text_report (ds_recordset set)`

Returns a formatted text report for the record set.

The report is returned as a single multi-line string.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the report. The caller is responsible for `free()` ing this pointer.

4.41.2.6 `ds_record ds_recordset_next_record (ds_recordset set)`

Returns the next record in the record set.

This function returns the "current record", and advances the current record pointer. Subsequent calls to this function will return successive records.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the next record, or `NULL` if the end of the record set has been reached.

4.41.2.7 `size_t ds_recordset_num_fields (ds_recordset set)`

Returns the number of fields in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of fields in the record set.

4.41.2.8 `size_t ds_recordset_num_records (ds_recordset set)`

Returns the number of records in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of records in the record set.

4.41.2.9 `void ds_recordset_seek_start (ds_recordset set)`

Sets the current record to the first record.

Parameters

<i>set</i>	The record set.
------------	-----------------

4.41.2.10 `void ds_recordset_set_headers (ds_recordset set, ds_record headers)`

Sets the record headers in a record set.

Parameters

<i>set</i>	The record set.
<i>headers</i>	The headers, in the form of a ds_record of strings. The list <i>must</i> have the same number of elements as the number of fields provided to ds_recordset_create() .

- Adds a record to a record set.*
- size_t `ds_recordset_num_fields` (`ds_recordset` set)
Returns the number of fields in a record set.
- size_t `ds_recordset_num_records` (`ds_recordset` set)
Returns the number of records in a record set.
- void `ds_recordset_set_headers` (`ds_recordset` set, `ds_record` headers)
Sets the record headers in a record set.
- `ds_str` `ds_recordset_get_text_report` (`ds_recordset` set)
Returns a formatted text report for the record set.
- `ds_str` `ds_recordset_get_next_insert_query` (`ds_recordset` set, const char *table_name)
Gets the next SQL INSERT query.
- void `ds_recordset_seek_start` (`ds_recordset` set)
Sets the current record to the first record.
- `ds_record` `ds_recordset_next_record` (`ds_recordset` set)
Returns the next record in the record set.

4.42.1 Detailed Description

Interface to record set structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.42.2 Typedef Documentation

4.42.2.1 typedef struct `ds_recordset*` `ds_recordset`

Typedef for opaque record set data type

4.42.3 Function Documentation

4.42.3.1 `ds_record` `ds_recordset_add_record` (`ds_recordset` set, `ds_record` record)

Adds a record to a record set.

The record *must* have the same number of fields as the number of fields provided to `ds_recordset_create()`.

Parameters

<i>set</i>	The record set to which to add.
<i>record</i>	The record to add.

Returns

A pointer to the new record (i.e. it returns the second parameter) or `NULL` on failure.

4.42.3.2 `ds_recordset ds_recordset_create (const size_t num_fields)`

Creates a new record set.

Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

Returns

A pointer to the new record set.

4.42.3.3 `void ds_recordset_destroy (ds_recordset set)`

Destroys a record set and frees associated resources.

Parameters

<i>set</i>	The record set to destroy.
------------	----------------------------

4.42.3.4 `ds_str ds_recordset_get_next_insert_query (ds_recordset set, const char * table_name)`

Gets the next SQL INSERT query.

Parameters

<i>set</i>	The set.
<i>table_name</i>	The table name into which to insert.

Returns

The query. Caller is responsible for `free()` ing.

4.42.3.5 `ds_str ds_recordset_get_text_report (ds_recordset set)`

Returns a formatted text report for the record set.

The report is returned as a single multi-line string.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the report. The caller is responsible for `free()` ing this pointer.

4.42.3.6 `ds_record ds_recordset_next_record (ds_recordset set)`

Returns the next record in the record set.

This function returns the "current record", and advances the current record pointer. Subsequent calls to this function will return successive records.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the next record, or `NULL` if the end of the record set has been reached.

4.42.3.7 `size_t ds_recordset_num_fields (ds_recordset set)`

Returns the number of fields in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of fields in the record set.

4.42.3.8 `size_t ds_recordset_num_records (ds_recordset set)`

Returns the number of records in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of records in the record set.

4.42.3.9 `void ds_recordset_seek_start (ds_recordset set)`

Sets the current record to the first record.

Parameters

<i>set</i>	The record set.
------------	-----------------

4.42.3.10 `void ds_recordset_set_headers (ds_recordset set, ds_record headers)`

Sets the record headers in a record set.

Parameters

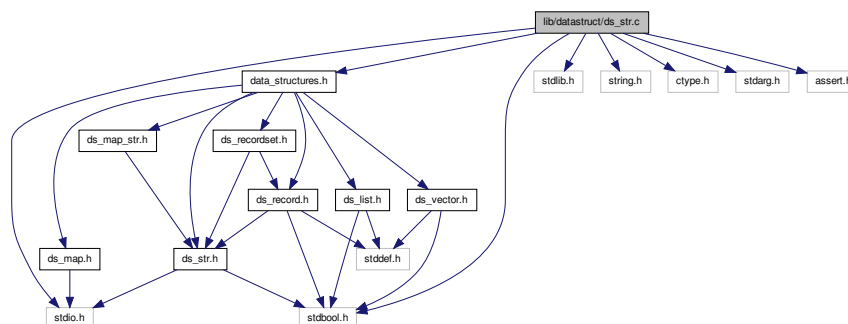
<i>set</i>	The record set.
<i>headers</i>	The headers, in the form of a ds_record of strings. The list <i>must</i> have the same number of elements as the number of fields provided to ds_recordset_create() .

4.43 lib/datastruct/ds_str.c File Reference

Implementation of string data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <assert.h>
#include "data_structures.h"
```

Include dependency graph for ds_str.c:



Data Structures

- struct [ds_str](#)

Functions

- [ds_str ds_str_create_direct](#) (char *init_str, const size_t init_str_size)
Creates a string using allocated memory.
- [ds_str ds_str_create](#) (const char *init_str)
Creates a new string from a C-style string.
- [ds_str ds_str_dup](#) (ds_str src)
Creates a new string from another string.
- [ds_str ds_str_create_sprintf](#) (const char *format,...)
Creates a string with `sprintf()`-type format.
- void [ds_str_destroy](#) (ds_str str)
Destroys a string and releases allocated resources.
- void [ds_str_destructor](#) (void *str)
Destroys a string and releases allocated resources.
- [ds_str ds_str_assign](#) (ds_str dst, ds_str src)
Assigns a string to another.
- [ds_str ds_str_assign_cstr](#) (ds_str dst, const char *src)
Assigns a C-style string to a string.
- const char * [ds_str_cstr](#) (ds_str str)
Returns a C-style string containing the string's contents.
- size_t [ds_str_length](#) (ds_str str)

- Returns the length of a string.*

 - `ds_str ds_str_size_to_fit (ds_str str)`

Reduces a string's capacity to fit its length.
- `ds_str ds_str_concat (ds_str dst, ds_str src)`

Concatenates two strings.
- `ds_str ds_str_concat_cstr (ds_str dst, const char *src)`

Concatenates a C-style string to a string.
- `ds_str ds_str_trunc (ds_str str, const size_t length)`

Truncates a string.
- unsigned long `ds_str_hash (ds_str str)`

Calculates a hash of a string.
- int `ds_str_compare (ds_str s1, ds_str s2)`

Compares two strings.
- int `ds_str_compare_cstr (ds_str s1, const char *s2)`

Compares a string with a C-style string.
- int `ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.
- `ds_str ds_str_substr_left (ds_str str, const size_t numchars)`

Returns a left substring.
- `ds_str ds_str_substr_right (ds_str str, const size_t numchars)`

Returns a right substring.
- void `ds_str_split (ds_str src, ds_str *left, ds_str *right, const char sc)`

Splits a string.
- void `ds_str_trim_leading (ds_str str)`

Trims leading whitespace in-place.
- void `ds_str_trim_trailing (ds_str str)`

Trims trailing whitespace in-place.
- void `ds_str_trim (ds_str str)`

Trims leading and trailing whitespace in-place.
- char `ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.
- bool `ds_str_is_empty (ds_str str)`

Checks if a string is empty.
- void `ds_str_clear (ds_str str)`

Clears (empties) a string.
- bool `ds_str_intval (ds_str str, const int base, int *value)`

Gets the integer value of a string.
- bool `ds_str_doubleval (ds_str str, double *value)`

Gets the double value of a string.
- `ds_str ds_str_getline (ds_str str, const size_t size, FILE *fp)`

Gets a line from a file and assigns it to a string.
- `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

4.43.1 Detailed Description

Implementation of string data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.43.2 Function Documentation

4.43.2.1 `ds_str ds_str_assign (ds_str dst, ds_str src)`

Assigns a string to another.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

Returns

dst on success, `NULL` on failure.

4.43.2.2 `ds_str ds_str_assign_cstr (ds_str dst, const char * src)`

Assigns a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

Returns

dst on success, `NULL` on failure.

4.43.2.3 `char ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.

Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

Returns

The character at the specified index.

4.43.2.4 void ds_str_clear (ds_str str)

Clears (empties) a string.

Parameters

<i>str</i>	The string.
------------	-------------

4.43.2.5 int ds_str_compare (ds_str s1, ds_str s2)

Compares two strings.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second string.

Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

4.43.2.6 int ds_str_compare_cstr (ds_str s1, const char * s2)

Compares a string with a C-style string.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

4.43.2.7 ds_str ds_str_concat (ds_str dst, ds_str src)

Concatenates two strings.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or NULL on failure.

4.43.2.8 ds_str ds_str_concat_cstr (ds_str dst, const char * src)

Concatenates a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

4.43.2.9 `ds_str ds_str_create (const char * init_str)`

Creates a new string from a C-style string.

Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

Returns

The new string, or `NULL` on failure.

4.43.2.10 `ds_str ds_str_create_direct (char * init_str, const size_t init_str_size)`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

Parameters

<i>init_str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

Returns

The new string, or `NULL` on failure.

4.43.2.11 `ds_str ds_str_create_sprintf (const char * format, ...)`

Creates a string with `sprintf()`-type format.

Parameters

<i>format</i>	The format string.
<code>...</code>	The subsequent arguments as specified by the format string.

Returns

The new string, or `NULL` on failure.

4.43.2.12 `const char* ds_str_cstr (ds_str str)`

Returns a C-style string containing the string's contents.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The C-style string containing the string's contents. The caller should not directly modify this string.

4.43.2.13 `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

Parameters

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or NULL to add <i>left_dec</i> to both sides.

Returns

The decorated string.

4.43.2.14 `void ds_str_destroy (ds_str str)`

Destroys a string and releases allocated resources.

Parameters

<i>str</i>	The string to destroy..
------------	-------------------------

4.43.2.15 `void ds_str_destructor (void * str)`

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

Parameters

<i>str</i>	The string to destroy.
------------	------------------------

4.43.2.16 `bool ds_str_doubleval (ds_str str, double * value)`

Gets the double value of a string.

Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid double value.

4.43.2.17 `ds_str ds_str_dup (ds_str src)`

Creates a new string from another string.

Parameters

<i>src</i>	The other string.
------------	-------------------

Returns

The new string, or `NULL` on failure.

4.43.2.18 `ds_str ds_str_getline (ds_str str, const size_t size, FILE * fp)`

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

Parameters

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

Returns

`dst`

4.43.2.19 `unsigned long ds_str_hash (ds_str str)`

Calculates a hash of a string.

Uses Dan Bernstein's djb2 algorithm.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The hash value

4.43.2.20 `bool ds_str_intval (ds_str str, const int base, int * value)`

Gets the integer value of a string.

Parameters

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .

<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.
--------------	--

Returns

`true` on successful conversion, `false` if the string does not contain a valid integer value.

4.43.2.21 bool ds_str_is_empty (ds_str str)

Checks if a string is empty.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

`true` if the string is empty, `false` otherwise.

4.43.2.22 size_t ds_str_length (ds_str str)

Returns the length of a string.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The length of the string.

4.43.2.23 ds_str ds_str_size_to_fit (ds_str str)

Reduces a string's capacity to fit its length.

Parameters

<i>str</i>	The string to size.
------------	---------------------

Returns

`str`, or `NULL` on failure.

4.43.2.24 void ds_str_split (ds_str src, ds_str * left, ds_str * right, const char sc)

Splits a string.

Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

4.43.2.25 `int ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.

Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

Returns

The index of the first occurrence, or -1 if the character was not found.

4.43.2.26 `ds_str ds_str_substr_left (ds_str str, const size_t numchars)`

Returns a left substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

4.43.2.27 `ds_str ds_str_substr_right (ds_str str, const size_t numchars)`

Returns a right substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

4.43.2.28 `void ds_str_trim (ds_str str)`

Trims leading and trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.43.2.29 void ds_str_trim_leading (ds_str str)

Trims leading whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.43.2.30 void ds_str_trim_trailing (ds_str str)

Trims trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.43.2.31 ds_str ds_str_trunc (ds_str str, const size_t length)

Truncates a string.

Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.

Returns

The original string, or `NULL` on failure.

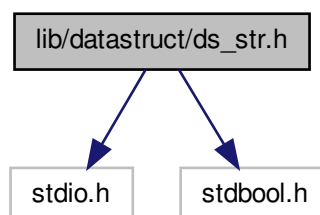
4.44 lib/datastruct/ds_str.h File Reference

Interface to string data structure.

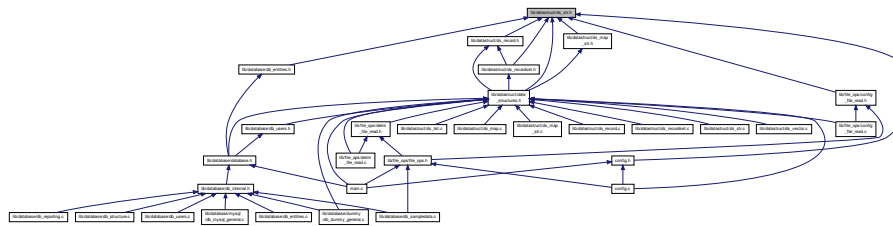
```
#include <stdio.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_str.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_str](#) * [ds_str](#)

Functions

- [ds_str ds_str_create](#) (const char *init_str)
Creates a new string from a C-style string.
- [ds_str ds_str_dup](#) (ds_str src)
Creates a new string from another string.
- [ds_str ds_str_create_sprintf](#) (const char *format,...)
Creates a string with `sprintf()`-type format.
- [ds_str ds_str_create_direct](#) (char *init_str, const size_t init_str_size)
Creates a string using allocated memory.
- void [ds_str_destroy](#) (ds_str str)
Destroys a string and releases allocated resources.
- void [ds_str_destructor](#) (void *str)
Destroys a string and releases allocated resources.
- [ds_str ds_str_assign](#) (ds_str dst, ds_str src)
Assigns a string to another.
- [ds_str ds_str_assign_cstr](#) (ds_str dst, const char *src)
Assigns a C-style string to a string.
- const char * [ds_str_cstr](#) (ds_str str)
Returns a C-style string containing the string's contents.
- size_t [ds_str_length](#) (ds_str str)
Returns the length of a string.
- [ds_str ds_str_size_to_fit](#) (ds_str str)
Reduces a string's capacity to fit its length.
- [ds_str ds_str_concat](#) (ds_str dst, ds_str src)
Concatenates two strings.
- [ds_str ds_str_concat_cstr](#) (ds_str dst, const char *src)
Concatenates a C-style string to a string.
- [ds_str ds_str_trunc](#) (ds_str str, const size_t length)
Truncates a string.
- unsigned long [ds_str_hash](#) (ds_str str)
Calculates a hash of a string.
- int [ds_str_compare](#) (ds_str s1, ds_str s2)
Compares two strings.
- int [ds_str_compare_cstr](#) (ds_str s1, const char *s2)
Compares a string with a C-style string.

- int `ds_str_strchr` (`ds_str` str, const char ch, const int start)
Returns index of first occurrence of a character.
- `ds_str ds_str_substr_left` (`ds_str` str, const size_t numchars)
Returns a left substring.
- `ds_str ds_str_substr_right` (`ds_str` str, const size_t numchars)
Returns a right substring.
- void `ds_str_split` (`ds_str` src, `ds_str` *left, `ds_str` *right, const char sc)
Splits a string.
- void `ds_str_trim_leading` (`ds_str` str)
Trims leading whitespace in-place.
- void `ds_str_trim_trailing` (`ds_str` str)
Trims trailing whitespace in-place.
- void `ds_str_trim` (`ds_str` str)
Trims leading and trailing whitespace in-place.
- char `ds_str_char_at_index` (`ds_str` str, const size_t index)
Returns the character at a specified index.
- bool `ds_str_is_empty` (`ds_str` str)
Checks if a string is empty.
- void `ds_str_clear` (`ds_str` str)
Clears (empties) a string.
- bool `ds_str_intval` (`ds_str` str, const int base, int *value)
Gets the integer value of a string.
- bool `ds_str_doubleval` (`ds_str` str, double *value)
Gets the double value of a string.
- `ds_str ds_str_getline` (`ds_str` str, const size_t size, FILE *fp)
Gets a line from a file and assigns it to a string.
- `ds_str ds_str_decorate` (`ds_str` str, `ds_str` left_dec, `ds_str` right_dec)
Brackets a string with decoration strings.

4.44.1 Detailed Description

Interface to string data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.44.2 Typedef Documentation

4.44.2.1 typedef struct ds_str* ds_str

Opaque data type for string

4.44.3 Function Documentation

4.44.3.1 `ds_str ds_str_assign (ds_str dst, ds_str src)`

Assigns a string to another.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

Returns

dst on success, `NULL` on failure.

4.44.3.2 `ds_str ds_str_assign_cstr (ds_str dst, const char * src)`

Assigns a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

Returns

dst on success, `NULL` on failure.

4.44.3.3 `char ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.

Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

Returns

The character at the specified index.

4.44.3.4 `void ds_str_clear (ds_str str)`

Clears (empties) a string.

Parameters

<i>str</i>	The string.
------------	-------------

4.44.3.5 `int ds_str_compare (ds_str s1, ds_str s2)`

Compares two strings.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second string.

Returns

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

4.44.3.6 int ds_str_compare_cstr (ds_str *s1*, const char * *s2*)

Compares a string with a C-style string.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

Returns

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

4.44.3.7 ds_str ds_str_concat (ds_str *dst*, ds_str *src*)

Concatenates two strings.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

4.44.3.8 ds_str ds_str_concat_cstr (ds_str *dst*, const char * *src*)

Concatenates a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

4.44.3.9 ds_str ds_str_create (const char * *init_str*)

Creates a new string from a C-style string.

Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

Returns

The new string, or `NULL` on failure.

4.44.3.10 `ds_str ds_str_create_direct (char * init_str, const size_t init_str_size)`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

Parameters

<i>init_str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

Returns

The new string, or `NULL` on failure.

4.44.3.11 `ds_str ds_str_create_sprintf (const char * format, ...)`

Creates a string with `sprintf()`-type format.

Parameters

<i>format</i>	The format string.
<i>...</i>	The subsequent arguments as specified by the format string.

Returns

The new string, or `NULL` on failure.

4.44.3.12 `const char* ds_str_cstr (ds_str str)`

Returns a C-style string containing the string's contents.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The C-style string containing the string's contents. The caller should not directly modify this string.

4.44.3.13 `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

Parameters

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or <code>NULL</code> to add <i>left_dec</i> to both sides.

Returns

The decorated string.

4.44.3.14 `void ds_str_destroy (ds_str str)`

Destroys a string and releases allocated resources.

Parameters

<i>str</i>	The string to destroy..
------------	-------------------------

4.44.3.15 `void ds_str_destructor (void * str)`

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

Parameters

<i>str</i>	The string to destroy.
------------	------------------------

4.44.3.16 `bool ds_str_doubleval (ds_str str, double * value)`

Gets the double value of a string.

Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid double value.

4.44.3.17 `ds_str ds_str_dup (ds_str src)`

Creates a new string from another string.

Parameters

<i>src</i>	The other string.
------------	-------------------

Returns

The new string, or `NULL` on failure.

4.44.3.18 `ds_str ds_str_getline (ds_str str, const size_t size, FILE * fp)`

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

Parameters

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

Returns

`dst`

4.44.3.19 `unsigned long ds_str_hash (ds_str str)`

Calculates a hash of a string.

Uses Dan Bernstein's djb2 algorithm.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The hash value

4.44.3.20 `bool ds_str_intval (ds_str str, const int base, int * value)`

Gets the integer value of a string.

Parameters

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .
<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid integer value.

4.44.3.21 `bool ds_str_is_empty (ds_str str)`

Checks if a string is empty.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

`true` is the string is empty, `false` otherwise.

4.44.3.22 `size_t ds_str_length (ds_str str)`

Returns the length of a string.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The length of the string.

4.44.3.23 `ds_str ds_str_size_to_fit (ds_str str)`

Reduces a string's capacity to fit its length.

Parameters

<i>str</i>	The string to size.
------------	---------------------

Returns

`str`, or `NULL` on failure.

4.44.3.24 `void ds_str_split (ds_str src, ds_str * left, ds_str * right, const char sc)`

Splits a string.

Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

4.44.3.25 `int ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.

Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

Returns

The index of the first occurrence, or -1 if the character was not found.

4.44.3.26 ds_str ds_str_substr_left (ds_str str, const size_t numchars)

Returns a left substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

4.44.3.27 ds_str ds_str_substr_right (ds_str str, const size_t numchars)

Returns a right substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

4.44.3.28 void ds_str_trim (ds_str str)

Trims leading and trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.44.3.29 void ds_str_trim_leading (ds_str str)

Trims leading whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.44.3.30 void ds_str_trim_trailing (ds_str str)

Trims trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

4.44.3.31 ds_str ds_str_trunc (ds_str str, const size_t length)

Truncates a string.

Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.

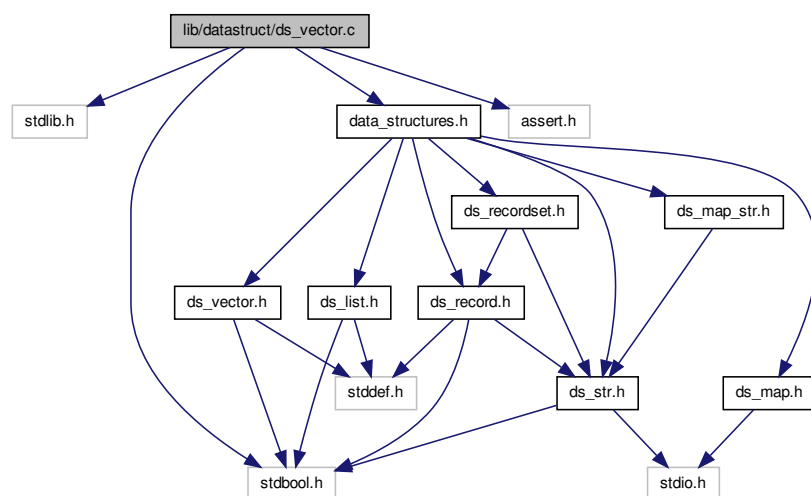
Returns

The original string, or `NULL` on failure.

4.45 lib/datastruct/ds_vector.c File Reference

Implementation of generic doubly-linked vector data structure.

```
#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_vector.c:
```



Data Structures

- struct [ds_vector](#)

Functions

- [ds_vector ds_vector_create](#) (const size_t size, const bool free_on_delete, void(*destructor)(void *))
Creates a new vector.
- void [ds_vector_destroy](#) ([ds_vector](#) vector)
Destroys a vector and frees any associated resources.
- void [ds_vector_destructor](#) (void *vector)
A vector destructor function.
- void [ds_vector_clear](#) ([ds_vector](#) vector)
Clears all the elements in a vector.
- void [ds_vector_set](#) ([ds_vector](#) vector, const size_t index, void *element)
Sets an element of a vector.
- void * [ds_vector_element](#) ([ds_vector](#) vector, const size_t index)
Retrieves the data at a specified index.
- size_t [ds_vector_size](#) ([ds_vector](#) vector)
Returns the size of a vector.
- void [ds_vector_seek_start](#) ([ds_vector](#) vector)
Sets the current element to the first element of a vector.
- void * [ds_vector_get_next_data](#) ([ds_vector](#) vector)
Returns the next element of the vector.

4.45.1 Detailed Description

Implementation of generic doubly-linked vector data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.45.2 Function Documentation

4.45.2.1 void ds_vector_clear (ds_vector vector)

Clears all the elements in a vector.

If the vector was created with `free_on_delete`, the elements are `free()`d prior to being cleared (i.e. set to NULL).

Parameters

<i>vector</i>	The vector.
---------------	-------------

4.45.2.2 `ds_vector ds_vector_create (const size_t size, const bool free_on_delete, void(*) (void *) destructor)`

Creates a new vector.

Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created vector, or `NULL` on failure.

4.45.2.3 `void ds_vector_destroy (ds_vector vector)`

Destroys a vector and frees any associated resources.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

4.45.2.4 `void ds_vector_destructor (void * vector)`

A vector destructor function.

This function may be passed to `ds_vector_create()` when creating a vector of vectors. It calls `ds_vector_destroy()`, but the parameter of `ds_vector_destroy()` is not compatible with the function signature expected by `ds_vector_create()`, so this function provides an appropriate interface.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

4.45.2.5 `void* ds_vector_element (ds_vector vector, const size_t index)`

Retrieves the data at a specified index.

Parameters

<i>vector</i>	The vector from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

4.45.2.6 `void* ds_vector_get_next_data (ds_vector vector)`

Returns the next element of the vector.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

A pointer to the next element, or `NULL` if the end of the vector has been reached.

4.45.2.7 void ds_vector_seek_start (ds_vector vector)

Sets the current element to the first element of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

4.45.2.8 void ds_vector_set (ds_vector vector, const size_t index, void * element)

Sets an element of a vector.

If the element is currently occupied, the existing element is `free()`d.

Parameters

<i>vector</i>	The vector to which to set.
<i>index</i>	The index of the element to set.
<i>element</i>	The element to set.

4.45.2.9 size_t ds_vector_size (ds_vector vector)

Returns the size of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

The size of the vector.

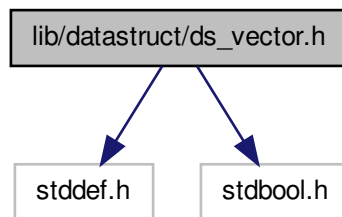
4.46 lib/datastruct/ds_vector.h File Reference

Interface to generic doubly-linked vector data structure.

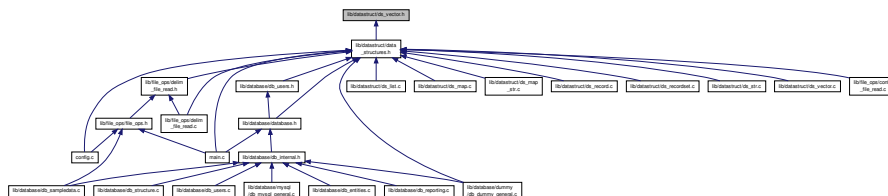
```
#include <stddef.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_vector.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_vector](#) * [ds_vector](#)

Functions

- [ds_vector ds_vector_create](#) (const [size_t](#) size, const bool free_on_delete, void(*destructor)(void *))
Creates a new vector.
- void [ds_vector_destroy](#) ([ds_vector](#) vector)
Destroys a vector and frees any associated resources.
- void [ds_vector_destructor](#) (void *vector)
A vector destructor function.
- void [ds_vector_clear](#) ([ds_vector](#) vector)
Clears all the elements in a vector.
- void [ds_vector_set](#) ([ds_vector](#) vector, const [size_t](#) index, void *element)
Sets an element of a vector.

- void * `ds_vector_element` (`ds_vector` vector, const size_t index)
Retrieves the data at a specified index.
- size_t `ds_vector_size` (`ds_vector` vector)
Returns the size of a vector.
- void `ds_vector_seek_start` (`ds_vector` vector)
Sets the current element to the first element of a vector.
- void * `ds_vector_get_next_data` (`ds_vector` vector)
Returns the next element of the vector.

4.46.1 Detailed Description

Interface to generic doubly-linked vector data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.46.2 Typedef Documentation

4.46.2.1 typedef struct ds_vector* ds_vector

Typedef for opaque vector datatype

4.46.3 Function Documentation

4.46.3.1 void ds_vector_clear (ds_vector vector)

Clears all the elements in a vector.

If the vector was created with `free_on_delete`, the elements are `free()`d prior to being cleared (i.e. set to NULL).

Parameters

<i>vector</i>	The vector.
---------------	-------------

4.46.3.2 ds_vector ds_vector_create (const size_t size, const bool free_on_delete, void(*) (void *) destructor)

Creates a new vector.

Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to NULL, <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created vector, or `NULL` on failure.

4.46.3.3 void ds_vector_destroy (ds_vector vector)

Destroys a vector and frees any associated resources.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

4.46.3.4 void ds_vector_destructor (void * vector)

A vector destructor function.

This function may be passed to `ds_vector_create()` when creating a vector of vectors. It calls `ds_vector_destroy()`, but the parameter of `ds_vector_destroy()` is not compatible with the function signature expected by `ds_vector_create()`, so this function provides an appropriate interface.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

4.46.3.5 void* ds_vector_element (ds_vector vector, const size_t index)

Retrieves the data at a specified index.

Parameters

<i>vector</i>	The vector from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

4.46.3.6 void* ds_vector_get_next_data (ds_vector vector)

Returns the next element of the vector.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

A pointer to the next element, or `NULL` if the end of the vector has been reached.

4.46.3.7 void ds_vector_seek_start (ds_vector vector)

Sets the current element to the first element of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

4.46.3.8 void ds_vector_set (ds_vector vector, const size_t index, void * element)

Sets an element of a vector.

If the element is currently occupied, the existing element is `free()`d.

Parameters

<i>vector</i>	The vector to which to set.
<i>index</i>	The index of the element to set.
<i>element</i>	The element to set.

4.46.3.9 size_t ds_vector_size (ds_vector vector)

Returns the size of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

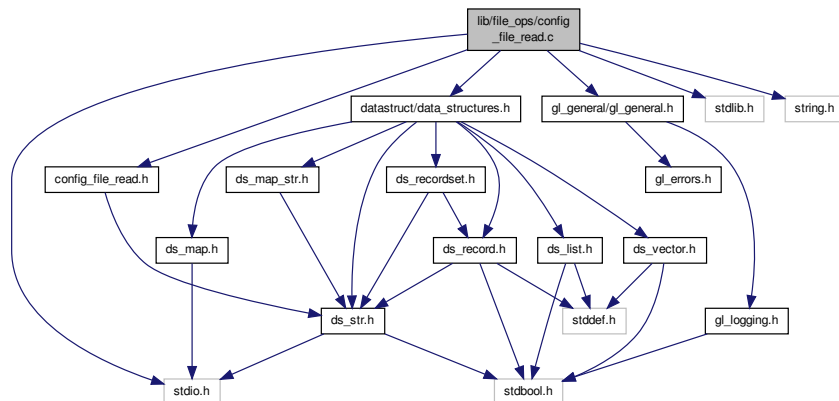
The size of the vector.

4.47 lib/file_ops/config_file_read.c File Reference

Implementation of configuration file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "config_file_read.h"
```

Include dependency graph for config_file_read.c:



Macros

- #define `MAX_BUFFER_SIZE` 1024
- #define `CONFIG_MAP_SIZE` 100

Functions

- int `config_file_read` (const char *filename)
Reads a configuration file and stores the key-value pairs.
- `ds_str config_file_value` (ds_str key)
Returns the value associated with a key.
- void `config_file_free` (void)
Frees the resources used by this module.

4.47.1 Detailed Description

Implementation of configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.47.2 Macro Definition Documentation

4.47.2.1 #define `CONFIG_MAP_SIZE` 100

Size to use for the hash map to contain the key-value pairs

4.47.2.2 `#define MAX_BUFFER_SIZE 1024`

Maximum size of buffers

4.47.3 Function Documentation

4.47.3.1 `void config_file_free (void)`

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if `config_file_read()` returned an error.

4.47.3.2 `int config_file_read (const char * filename)`

Reads a configuration file and stores the key-value pairs.

Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

Returns

CONFIG_FILE_OK on success, CONFIG_FILE_NO_FILE if the specified file could not be opened for reading, CONFIG_FILE_MALFORMED_FILE if the configuration file was improperly formed.

4.47.3.3 `ds_str config_file_value (ds_str key)`

Returns the value associated with a key.

Parameters

<i>key</i>	The specified key.
------------	--------------------

Returns

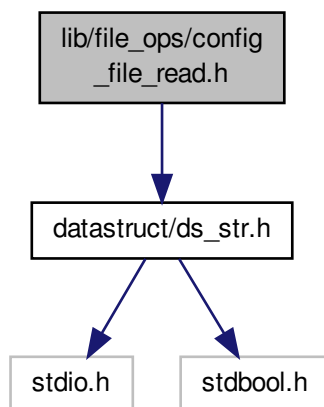
A pointer to the associated value, or `NULL` if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

4.48 `lib/file_ops/config_file_read.h` File Reference

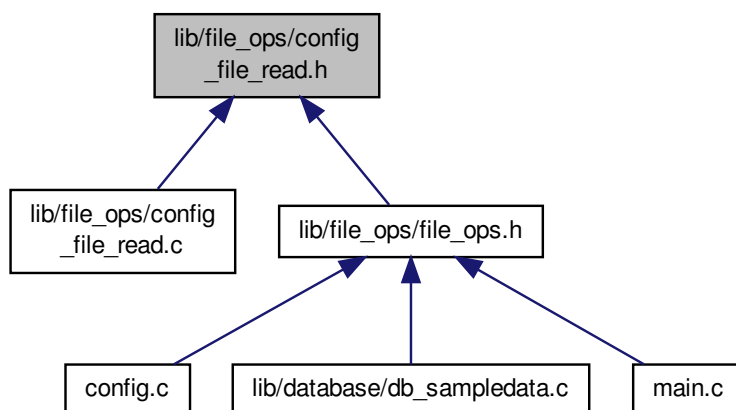
Interface to configuration file reading functionality.

```
#include "datastruct/ds_str.h"
```

Include dependency graph for config_file_read.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define CONFIG_FILE_OK 0`
- `#define CONFIG_FILE_NO_FILE 1`
- `#define CONFIG_FILE_MALFORMED_FILE 2`

Functions

- `int config_file_read (const char *filename)`

Reads a configuration file and stores the key-value pairs.

- void [config_file_free](#) (void)

Frees the resources used by this module.

- [ds_str config_file_value](#) ([ds_str](#) key)

Returns the value associated with a key.

4.48.1 Detailed Description

Interface to configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.48.2 Macro Definition Documentation

4.48.2.1 #define CONFIG_FILE_MALFORMED_FILE 2

Return status when configuration file is improperly formed

4.48.2.2 #define CONFIG_FILE_NO_FILE 1

Return status when unable to open file for reading

4.48.2.3 #define CONFIG_FILE_OK 0

Return status for success

4.48.3 Function Documentation

4.48.3.1 void config_file_free (void)

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if [config_file_read\(\)](#) returned an error.

4.48.3.2 int config_file_read (const char * filename)

Reads a configuration file and stores the key-value pairs.

Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

Returns

CONFIG_FILE_OK on success, CONFIG_FILE_NO_FILE if the specified file could not be opened for reading, CONFIG_FILE_MALFORMED_FILE if the configuration file was improperly formed.

4.48.3.3 ds_str config_file_value (ds_str key)

Returns the value associated with a key.

Parameters

<i>key</i>	The specified key.
------------	--------------------

Returns

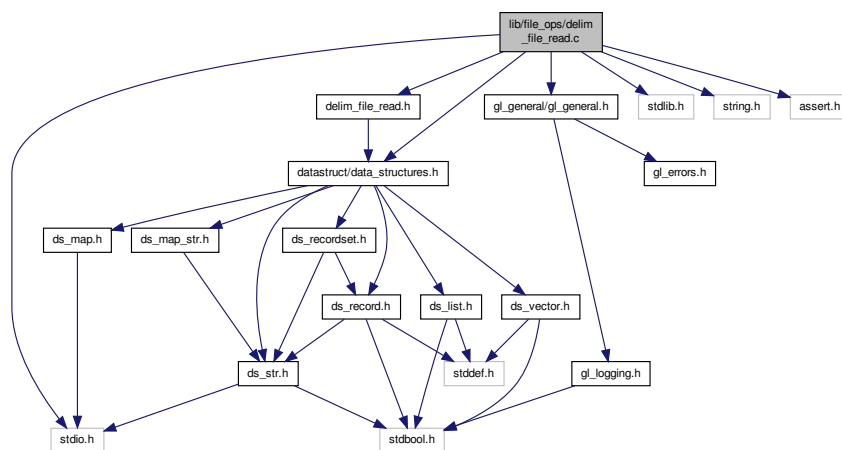
A pointer to the associated value, or NULL if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

4.49 lib/file_ops/delim_file_read.c File Reference

Implementation of delimited file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "delim_file_read.h"
```

Include dependency graph for delim_file_read.c:

**Macros**

- #define MAX_LINE_SIZE 1024

Functions

- [ds_recordset delim_file_read](#) (const char *filename, const char delim)
Constructs a [ds_recordset](#) from a delimited file.

4.49.1 Detailed Description

Implementation of delimited file reading functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.49.2 Macro Definition Documentation

4.49.2.1 #define MAX_LINE_SIZE 1024

Maximum size of buffers

4.49.3 Function Documentation

4.49.3.1 ds_recordset delim_file_read (const char * filename, const char delim)

Constructs a [ds_recordset](#) from a delimited file.

Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

Returns

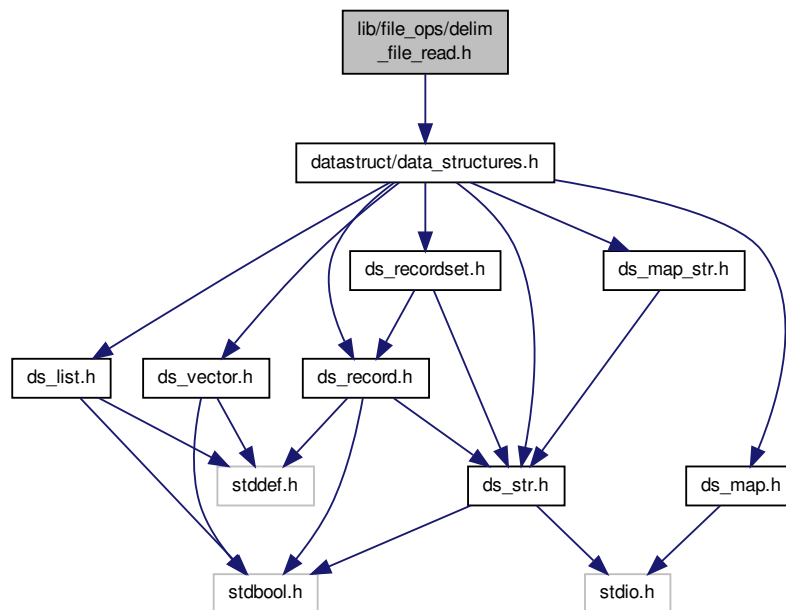
The [ds_recordset](#), or NULL on failure.

4.50 lib/file_ops/delim_file_read.h File Reference

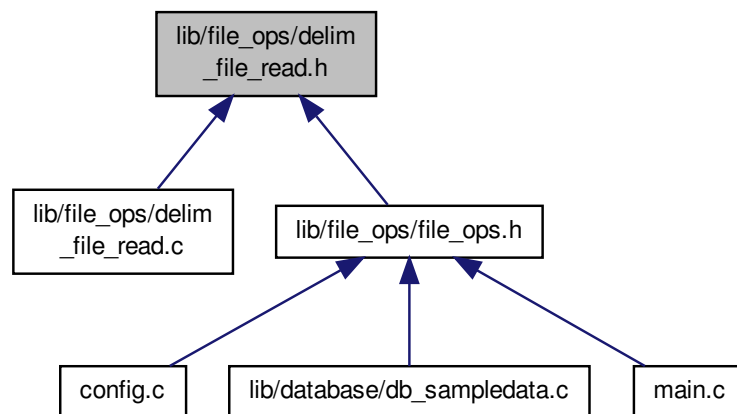
Interface to delimited file reading functionality.

```
#include "datastruct/data_structures.h"
```

Include dependency graph for delim_file_read.h:



This graph shows which files directly or indirectly include this file:



Functions

- [ds_recordset delim_file_read](#) (const char *filename, const char delim)

Constructs a [ds_recordset](#) from a delimited file.

4.50.1 Detailed Description

Interface to delimited file reading functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.50.2 Function Documentation

4.50.2.1 `ds_recordset delim_file_read (const char * filename, const char delim)`

Constructs a `ds_recordset` from a delimited file.

Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

Returns

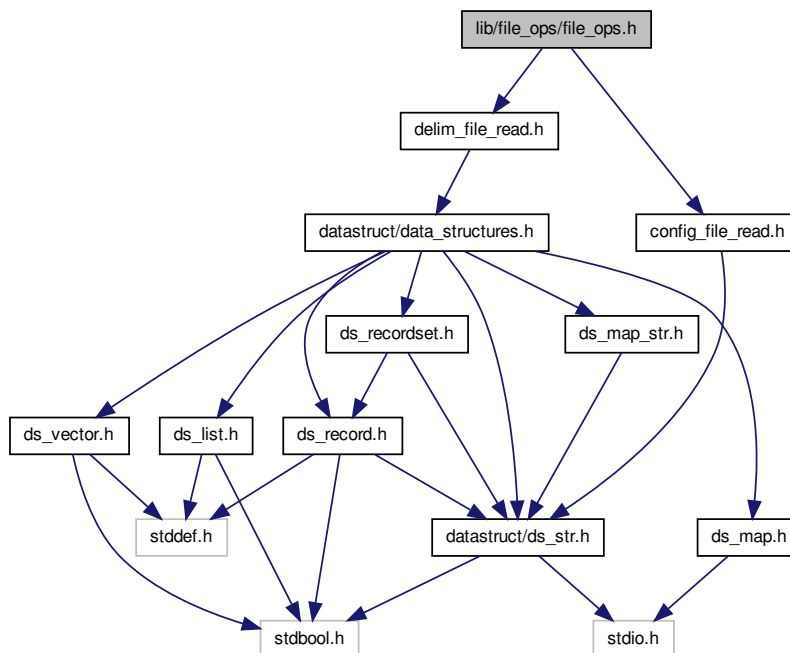
The `ds_recordset`, or `NULL` on failure.

4.51 `lib/file_ops/file_ops.h` File Reference

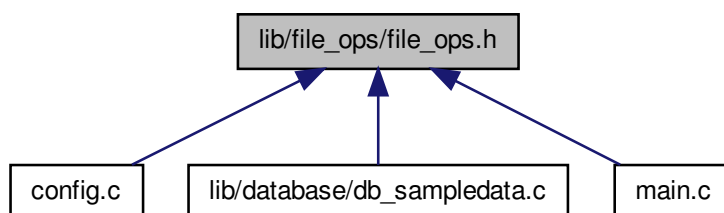
User interface to file operations functionality.

```
#include "config_file_read.h"
#include "delim_file_read.h"
```

Include dependency graph for file_ops.h:



This graph shows which files directly or indirectly include this file:



4.51.1 Detailed Description

User interface to file operations functionality.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

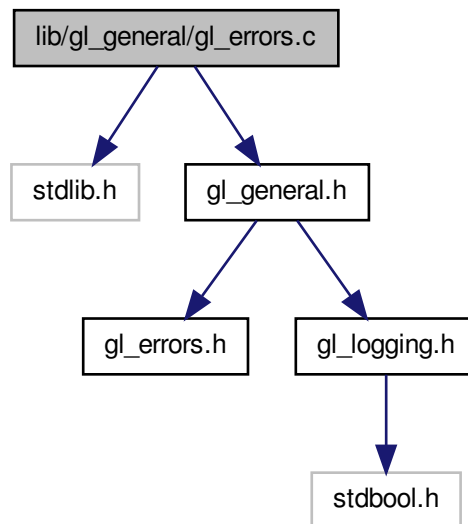
4.52 lib/gl_general/gl_errors.c File Reference

Implementation of error functionality.

```
#include <stdlib.h>
```

```
#include "gl_general.h"
```

Include dependency graph for gl_errors.c:



Functions

- void `gl_error_quit` (const char *msg)
Logs an error message and quits program.

4.52.1 Detailed Description

Implementation of error functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.52.2 Function Documentation

4.52.2.1 void `gl_error_quit` (const char * msg)

Logs an error message and quits program.

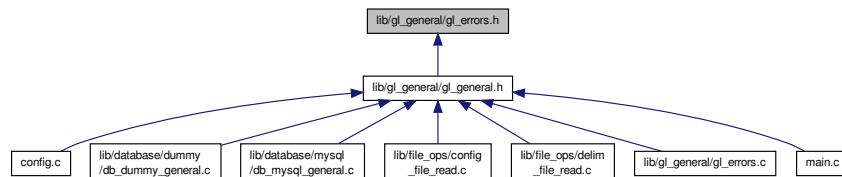
Parameters

<i>msg</i>	The error message to log.
------------	---------------------------

4.53 lib/gl_general/gl_errors.h File Reference

Interface to error functionality.

This graph shows which files directly or indirectly include this file:



Functions

- void [gl_error_quit](#) (const char *msg)
Logs an error message and quits program.

4.53.1 Detailed Description

Interface to error functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.53.2 Function Documentation

4.53.2.1 void gl_error_quit (const char * msg)

Logs an error message and quits program.

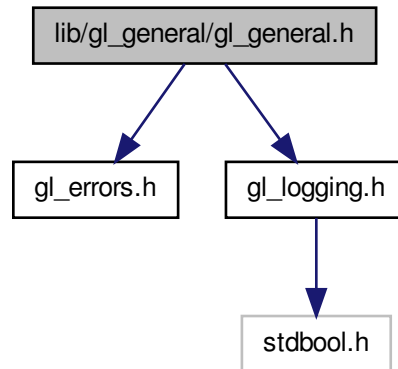
Parameters

<i>msg</i>	The error message to log.
------------	---------------------------

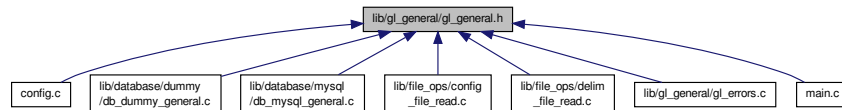
4.54 lib/gl_general/gl_general.h File Reference

User interface to logging and error functionality.

```
#include "gl_errors.h"
#include "gl_logging.h"
Include dependency graph for gl_general.h:
```



This graph shows which files directly or indirectly include this file:



4.54.1 Detailed Description

User interface to logging and error functionality.

Author

Paul Griffiths

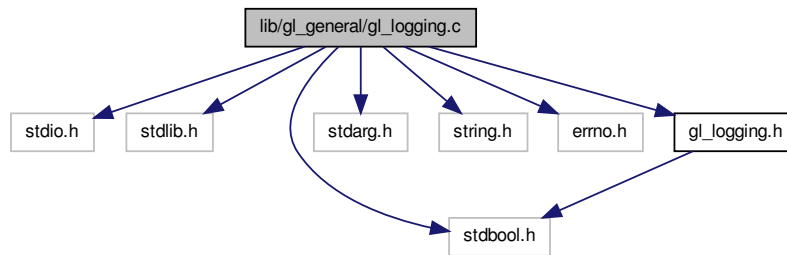
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.55 lib/gl_general/gl_logging.c File Reference

Implementation of logging functionality.


```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include "gl_logging.h"
Include dependency graph for gl_logging.c:
```



Functions

- void [gl_set_logging](#) (const bool status)
Turns logging on or off.
- void [gl_log_msg](#) (const char *format,...)
Logs a message to the log file.

4.55.1 Detailed Description

Implementation of logging functionality. Implementation of logging functionality. Enables debugging and other system messages to be recorded to a log file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.55.2 Function Documentation

4.55.2.1 void [gl_log_msg](#) (const char * *format*, ...)

Logs a message to the log file.

Logs a message to the log file.

Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
...	Variable arguments as specified by format string.

4.55.2.2 void gl_set_logging (const bool status)

Turns logging on or off.

Turns logging on or off.

Parameters

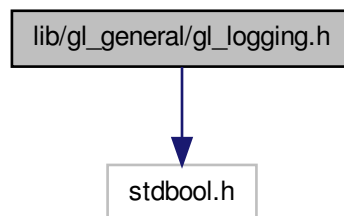
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

4.56 lib/gl_general/gl_logging.h File Reference

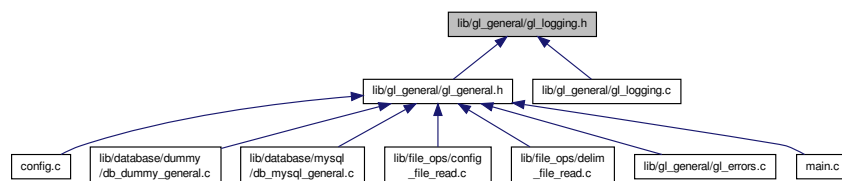
Interface to logging functionality.

```
#include <stdbool.h>
```

Include dependency graph for gl_logging.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [gl_set_logging](#) (const bool status)
Turns logging on or off.
- void [gl_log_msg](#) (const char *format,...)
Logs a message to the log file.

4.56.1 Detailed Description

Interface to logging functionality. Interface to logging functionality. Enables debugging and other system messages to be recorded to a log file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.56.2 Function Documentation**4.56.2.1 void gl_log_msg (const char * *format*, ...)**

Logs a message to the log file.

Logs a message to the log file.

Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
...	Variable arguments as specified by format string.

4.56.2.2 void gl_set_logging (const bool *status*)

Turns logging on or off.

Turns logging on or off.

Parameters

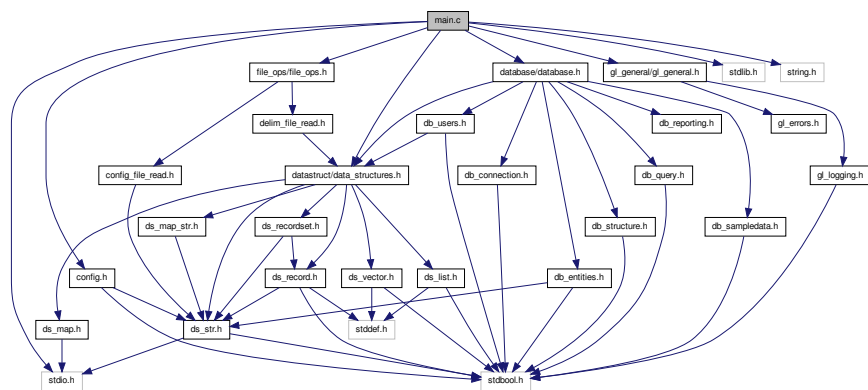
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

4.57 main.c File Reference

Main function for `general_ledger`.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/database.h"
#include "config.h"
#include "datastruct/data_structures.h"
#include "file_ops/file_ops.h"
```

Include dependency graph for main.c:



Functions

- `ds_str login` (void)
Logs a user in and retrieves the password.
- void `print_usage_message` (char *progrname)
Prints a program usage message.
- void `print_version_message` (char *progrname)
Prints a program version message.
- void `print_help_message` (char *progrname)
Prints a program help message.
- void `test_functionality` (void)
Casual test function.
- int `main` (int argc, char **argv)
Main function.

4.57.1 Detailed Description

Main function for general_ledger. Main function for general_ledger.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

4.57.2 Function Documentation

4.57.2.1 `ds_str login` (void)

Logs a user in and retrieves the password.

Returns

The password.

4.57.2.2 int main (int *argc*, char ** *argv*)

Main function.

Main function.

Returns

Exit status.

4.57.2.3 void print_help_message (char * *progrname*)

Prints a program help message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

4.57.2.4 void print_usage_message (char * *progrname*)

Prints a program usage message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

4.57.2.5 void print_version_message (char * *progrname*)

Prints a program version message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

4.57.2.6 void test_functionality (void)

Casual test function.

Used for casually testing program functionality.

Index

- `_XOPEN_SOURCE`
 - `config.c`, [20](#)
 - `db_dummy_general.c`, [47](#)
- `CONFIG_FILE_OK`
 - `config_file_read.h`, [120](#)
- `CONFIG_MAP_SIZE`
 - `config_file_read.c`, [117](#)
- `capacity`
 - `ds_str`, [13](#)
- `config.c`, [19](#)
 - `_XOPEN_SOURCE`, [20](#)
 - `get_cmdline_options`, [20](#)
 - `get_configuration`, [20](#)
 - `params_free`, [21](#)
 - `params_init`, [21](#)
- `config.h`, [21](#)
 - `get_cmdline_options`, [22](#)
 - `get_configuration`, [23](#)
 - `params_free`, [23](#)
 - `params_init`, [23](#)
- `config_file_free`
 - `config_file_read.c`, [118](#)
 - `config_file_read.h`, [120](#)
- `config_file_read`
 - `config_file_read.c`, [118](#)
 - `config_file_read.h`, [120](#)
- `config_file_read.c`
 - `CONFIG_MAP_SIZE`, [117](#)
 - `config_file_free`, [118](#)
 - `config_file_read`, [118](#)
 - `config_file_value`, [118](#)
 - `MAX_BUFFER_SIZE`, [117](#)
- `config_file_read.h`
 - `CONFIG_FILE_OK`, [120](#)
 - `config_file_free`, [120](#)
 - `config_file_read`, [120](#)
 - `config_file_value`, [121](#)
- `config_file_value`
 - `config_file_read.c`, [118](#)
 - `config_file_read.h`, [121](#)
- `conn_mss`
 - `db_mysql_general.c`, [53](#)
- `create`
 - `params`, [16](#)
- `current`
 - `ds_list`, [7](#)
 - `ds_vector`, [14](#)
- `data`
 - `ds_list_element`, [9](#)
 - `ds_str`, [13](#)
 - `ds_vector`, [14](#)
- `data_destructor`
 - `ds_list`, [8](#)
 - `ds_vector`, [14](#)
- `database`
 - `params`, [16](#)
- `db_connect`
 - `db_connection.h`, [26](#)
 - `db_dummy_general.c`, [47](#)
 - `db_mysql_general.c`, [53](#)
- `db_connection.h`
 - `db_connect`, [26](#)
- `db_create_database_structure`
 - `db_structure.c`, [38](#)
 - `db_structure.h`, [40](#)
- `db_create_entities_table`
 - `db_entities.c`, [27](#)
 - `db_entities.h`, [29](#)
- `db_create_entities_table_sql`
 - `db_dummy_create_entities_table_sql.c`, [44](#)
 - `db_mysql_create_entities_table_sql.c`, [49](#)
 - `db_sql.h`, [36](#)
- `db_create_recordset_from_query`
 - `db_dummy_general.c`, [47](#)
 - `db_mysql_general.c`, [53](#)
 - `db_reporting.h`, [33](#)
- `db_create_report_from_query`
 - `db_reporting.c`, [32](#)
 - `db_reporting.h`, [33](#)
- `db_create_users_table`
 - `db_users.c`, [41](#)
 - `db_users.h`, [43](#)
- `db_create_users_table_sql`
 - `db_dummy_create_users_table_sql.c`, [44](#)
 - `db_mysql_create_users_table_sql.c`, [50](#)
 - `db_sql.h`, [36](#)
- `db_delete_database_structure`
 - `db_structure.c`, [38](#)
 - `db_structure.h`, [40](#)
- `db_drop_entities_table`
 - `db_entities.c`, [27](#)
 - `db_entities.h`, [29](#)
- `db_drop_entities_table_sql`
 - `db_dummy_drop_entities_table_sql.c`, [45](#)
 - `db_mysql_drop_entities_table_sql.c`, [51](#)
 - `db_sql.h`, [37](#)
- `db_drop_users_table`

- db_users.c, 41
- db_users.h, 43
- db_drop_users_table_sql
 - db_dummy_drop_users_table_sql.c, 45
 - db_mysql_drop_users_table_sql.c, 51
 - db_sql.h, 37
- db_dummy_create_entities_table_sql.c
 - db_create_entities_table_sql, 44
- db_dummy_create_users_table_sql.c
 - db_create_users_table_sql, 44
- db_dummy_drop_entities_table_sql.c
 - db_drop_entities_table_sql, 45
- db_dummy_drop_users_table_sql.c
 - db_drop_users_table_sql, 45
- db_dummy_general.c
 - _XOPEN_SOURCE, 47
 - db_connect, 47
 - db_create_recordset_from_query, 47
 - db_execute_query, 47
- db_dummy_list_entities_report_sql.c
 - db_list_entities_report_sql, 48
- db_dummy_list_users_report_sql.c
 - db_list_users_report_sql, 49
- db_entities.c
 - db_create_entities_table, 27
 - db_drop_entities_table, 27
 - db_list_entities_report, 27
- db_entities.h
 - db_create_entities_table, 29
 - db_drop_entities_table, 29
 - db_list_entities_report, 29
- db_execute_query
 - db_dummy_general.c, 47
 - db_mysql_general.c, 53
 - db_query.h, 31
- db_list_entities_report
 - db_entities.c, 27
 - db_entities.h, 29
- db_list_entities_report_sql
 - db_dummy_list_entities_report_sql.c, 48
 - db_mysql_list_entities_report_sql.c, 54
 - db_sql.h, 37
- db_list_users_report
 - db_users.c, 41
 - db_users.h, 43
- db_list_users_report_sql
 - db_dummy_list_users_report_sql.c, 49
 - db_mysql_list_users_report_sql.c, 55
 - db_sql.h, 37
- db_mysql_create_entities_table_sql.c
 - db_create_entities_table_sql, 49
- db_mysql_create_users_table_sql.c
 - db_create_users_table_sql, 50
- db_mysql_drop_entities_table_sql.c
 - db_drop_entities_table_sql, 51
- db_mysql_drop_users_table_sql.c
 - db_drop_users_table_sql, 51
- db_mysql_general.c
 - conn_mss, 53
 - db_connect, 53
 - db_create_recordset_from_query, 53
 - db_execute_query, 53
 - main_mss, 53
- db_mysql_list_entities_report_sql.c
 - db_list_entities_report_sql, 54
- db_mysql_list_users_report_sql.c
 - db_list_users_report_sql, 55
- db_query.h
 - db_execute_query, 31
- db_reporting.c
 - db_create_report_from_query, 32
- db_reporting.h
 - db_create_recordset_from_query, 33
 - db_create_report_from_query, 33
- db_sql.h
 - db_create_entities_table_sql, 36
 - db_create_users_table_sql, 36
 - db_drop_entities_table_sql, 37
 - db_drop_users_table_sql, 37
 - db_list_entities_report_sql, 37
 - db_list_users_report_sql, 37
- db_structure.c
 - db_create_database_structure, 38
 - db_delete_database_structure, 38
- db_structure.h
 - db_create_database_structure, 40
 - db_delete_database_structure, 40
- db_users.c
 - db_create_users_table, 41
 - db_drop_users_table, 41
 - db_list_users_report, 41
- db_users.h
 - db_create_users_table, 43
 - db_drop_users_table, 43
 - db_list_users_report, 43
- delete_data
 - params, 16
- delim_file_read
 - delim_file_read.c, 122
 - delim_file_read.h, 124
- delim_file_read.c
 - delim_file_read, 122
 - MAX_LINE_SIZE, 122
- delim_file_read.h
 - delim_file_read, 124
- ds_list, 7
 - current, 7
 - data_destructor, 8
 - ds_list.h, 62
 - free_on_delete, 8
 - head, 8
 - length, 8
 - tail, 8
- ds_list.c
 - ds_list_append, 57
 - ds_list_create, 57

- ds_list_destroy, 58
- ds_list_destructor, 58
- ds_list_element, 58
- ds_list_get_next_data, 58
- ds_list_get_prev_data, 59
- ds_list_is_empty, 59
- ds_list_length, 59
- ds_list_remove_all, 59
- ds_list_remove_tail, 60
- ds_list_seek_end, 60
- ds_list_seek_start, 60
- ds_list.h
 - ds_list, 62
 - ds_list_append, 62
 - ds_list_create, 62
 - ds_list_destroy, 62
 - ds_list_destructor, 62
 - ds_list_element, 63
 - ds_list_get_next_data, 63
 - ds_list_get_prev_data, 63
 - ds_list_is_empty, 63
 - ds_list_length, 64
 - ds_list_remove_all, 64
 - ds_list_remove_tail, 64
 - ds_list_seek_end, 64
 - ds_list_seek_start, 64
- ds_list_append
 - ds_list.c, 57
 - ds_list.h, 62
- ds_list_create
 - ds_list.c, 57
 - ds_list.h, 62
- ds_list_destroy
 - ds_list.c, 58
 - ds_list.h, 62
- ds_list_destructor
 - ds_list.c, 58
 - ds_list.h, 62
- ds_list_element, 8
 - data, 9
 - ds_list.c, 58
 - ds_list.h, 63
 - next, 9
 - previous, 9
- ds_list_get_next_data
 - ds_list.c, 58
 - ds_list.h, 63
- ds_list_get_prev_data
 - ds_list.c, 59
 - ds_list.h, 63
- ds_list_is_empty
 - ds_list.c, 59
 - ds_list.h, 63
- ds_list_length
 - ds_list.c, 59
 - ds_list.h, 64
- ds_list_remove_all
 - ds_list.c, 59
- ds_list.h, 64
- ds_list_remove_tail
 - ds_list.c, 60
 - ds_list.h, 64
- ds_list_seek_end
 - ds_list.c, 60
 - ds_list.h, 64
- ds_list_seek_start
 - ds_list.c, 60
 - ds_list.h, 64
- ds_map, 9
 - ds_map.h, 68
 - hash_size, 10
 - lists, 10
- ds_map.c
 - ds_map_destroy, 66
 - ds_map_get_value, 66
 - ds_map_init, 66
 - ds_map_insert, 67
 - ds_map_print_all, 67
- ds_map.h
 - ds_map, 68
 - ds_map_destroy, 68
 - ds_map_get_value, 69
 - ds_map_init, 69
 - ds_map_insert, 69
 - ds_map_print_all, 69
- ds_map_destroy
 - ds_map.c, 66
 - ds_map.h, 68
- ds_map_get_value
 - ds_map.c, 66
 - ds_map.h, 69
- ds_map_init
 - ds_map.c, 66
 - ds_map.h, 69
- ds_map_insert
 - ds_map.c, 67
 - ds_map.h, 69
- ds_map_print_all
 - ds_map.c, 67
 - ds_map.h, 69
- ds_map_str, 10
 - ds_map_str.h, 73
 - hash_size, 10
 - lists, 11
- ds_map_str.c
 - ds_map_str_destroy, 71
 - ds_map_str_get_value, 71
 - ds_map_str_init, 71
 - ds_map_str_insert, 71
- ds_map_str.h
 - ds_map_str, 73
 - ds_map_str_destroy, 73
 - ds_map_str_get_value, 73
 - ds_map_str_init, 73
 - ds_map_str_insert, 74
- ds_map_str_destroy

- ds_map_str.c, 71
 - ds_map_str.h, 73
- ds_map_str_get_value
 - ds_map_str.c, 71
 - ds_map_str.h, 73
- ds_map_str_init
 - ds_map_str.c, 71
 - ds_map_str.h, 73
- ds_map_str_insert
 - ds_map_str.c, 71
 - ds_map_str.h, 74
- ds_record, 11
 - ds_record.h, 79
 - fields, 11
- ds_record.c
 - ds_record_clear, 75
 - ds_record_create, 75
 - ds_record_destroy, 76
 - ds_record_destructor, 76
 - ds_record_get_field, 76
 - ds_record_get_next_data, 76
 - ds_record_make_delim_string, 76
 - ds_record_make_values_string, 77
 - ds_record_seek_start, 77
 - ds_record_set_field, 77
 - ds_record_size, 77
 - ds_record_tokenize, 77
- ds_record.h
 - ds_record, 79
 - ds_record_clear, 79
 - ds_record_create, 80
 - ds_record_destroy, 80
 - ds_record_destructor, 80
 - ds_record_get_field, 80
 - ds_record_get_next_data, 80
 - ds_record_make_delim_string, 81
 - ds_record_make_values_string, 81
 - ds_record_seek_start, 81
 - ds_record_set_field, 81
 - ds_record_size, 81
 - ds_record_tokenize, 82
- ds_record_clear
 - ds_record.c, 75
 - ds_record.h, 79
- ds_record_create
 - ds_record.c, 75
 - ds_record.h, 80
- ds_record_destroy
 - ds_record.c, 76
 - ds_record.h, 80
- ds_record_destructor
 - ds_record.c, 76
 - ds_record.h, 80
- ds_record_get_field
 - ds_record.c, 76
 - ds_record.h, 80
- ds_record_get_next_data
 - ds_record.c, 76
- ds_record.h, 80
 - ds_record.h, 80
- ds_record_make_delim_string
 - ds_record.c, 76
 - ds_record.h, 81
- ds_record_make_values_string
 - ds_record.c, 77
 - ds_record.h, 81
- ds_record_seek_start
 - ds_record.c, 77
 - ds_record.h, 81
- ds_record_set_field
 - ds_record.c, 77
 - ds_record.h, 81
- ds_record_size
 - ds_record.c, 77
 - ds_record.h, 81
- ds_record_tokenize
 - ds_record.c, 77
 - ds_record.h, 82
- ds_recordset, 12
 - ds_recordset.h, 87
 - field_lengths, 12
 - headers, 12
 - num_fields, 12
 - records, 12
- ds_recordset.c
 - ds_recordset_add_record, 83
 - ds_recordset_create, 84
 - ds_recordset_destroy, 84
 - ds_recordset_get_next_insert_query, 84
 - ds_recordset_get_text_report, 84
 - ds_recordset_next_record, 84
 - ds_recordset_num_fields, 85
 - ds_recordset_num_records, 85
 - ds_recordset_seek_start, 85
 - ds_recordset_set_headers, 85
- ds_recordset.h
 - ds_recordset, 87
 - ds_recordset_add_record, 87
 - ds_recordset_create, 87
 - ds_recordset_destroy, 88
 - ds_recordset_get_next_insert_query, 88
 - ds_recordset_get_text_report, 88
 - ds_recordset_next_record, 88
 - ds_recordset_num_fields, 89
 - ds_recordset_num_records, 89
 - ds_recordset_seek_start, 89
 - ds_recordset_set_headers, 89
- ds_recordset_add_record
 - ds_recordset.c, 83
 - ds_recordset.h, 87
- ds_recordset_create
 - ds_recordset.c, 84
 - ds_recordset.h, 87
- ds_recordset_destroy
 - ds_recordset.c, 84
 - ds_recordset.h, 88
- ds_recordset_get_next_insert_query

- ds_recordset.c, 84
 - ds_recordset.h, 88
- ds_recordset_get_text_report
 - ds_recordset.c, 84
 - ds_recordset.h, 88
- ds_recordset_next_record
 - ds_recordset.c, 84
 - ds_recordset.h, 88
- ds_recordset_num_fields
 - ds_recordset.c, 85
 - ds_recordset.h, 89
- ds_recordset_num_records
 - ds_recordset.c, 85
 - ds_recordset.h, 89
- ds_recordset_seek_start
 - ds_recordset.c, 85
 - ds_recordset.h, 89
- ds_recordset_set_headers
 - ds_recordset.c, 85
 - ds_recordset.h, 89
- ds_str, 13
 - capacity, 13
 - data, 13
 - ds_str.h, 101
 - length, 13
- ds_str.c
 - ds_str_assign, 92
 - ds_str_assign_cstr, 92
 - ds_str_char_at_index, 92
 - ds_str_clear, 92
 - ds_str_compare, 93
 - ds_str_compare_cstr, 93
 - ds_str_concat, 93
 - ds_str_concat_cstr, 93
 - ds_str_create, 94
 - ds_str_create_direct, 94
 - ds_str_create_sprintf, 94
 - ds_str_cstr, 94
 - ds_str_decorate, 95
 - ds_str_destroy, 95
 - ds_str_destructor, 95
 - ds_str_doubleval, 95
 - ds_str_dup, 96
 - ds_str_getline, 96
 - ds_str_hash, 96
 - ds_str_intval, 96
 - ds_str_is_empty, 97
 - ds_str_length, 97
 - ds_str_size_to_fit, 97
 - ds_str_split, 97
 - ds_str_strchr, 98
 - ds_str_substr_left, 98
 - ds_str_substr_right, 98
 - ds_str_trim, 98
 - ds_str_trim_leading, 98
 - ds_str_trim_trailing, 99
 - ds_str_trunc, 99
- ds_str.h
 - ds_str, 101
 - ds_str_assign, 102
 - ds_str_assign_cstr, 102
 - ds_str_char_at_index, 102
 - ds_str_clear, 102
 - ds_str_compare, 102
 - ds_str_compare_cstr, 103
 - ds_str_concat, 103
 - ds_str_concat_cstr, 103
 - ds_str_create, 103
 - ds_str_create_direct, 104
 - ds_str_create_sprintf, 104
 - ds_str_cstr, 104
 - ds_str_decorate, 105
 - ds_str_destroy, 105
 - ds_str_destructor, 105
 - ds_str_doubleval, 105
 - ds_str_dup, 105
 - ds_str_getline, 106
 - ds_str_hash, 106
 - ds_str_intval, 106
 - ds_str_is_empty, 106
 - ds_str_length, 107
 - ds_str_size_to_fit, 107
 - ds_str_split, 107
 - ds_str_strchr, 107
 - ds_str_substr_left, 108
 - ds_str_substr_right, 108
 - ds_str_trim, 108
 - ds_str_trim_leading, 108
 - ds_str_trim_trailing, 109
 - ds_str_trunc, 109
- ds_str_assign
 - ds_str.c, 92
 - ds_str.h, 102
- ds_str_assign_cstr
 - ds_str.c, 92
 - ds_str.h, 102
- ds_str_char_at_index
 - ds_str.c, 92
 - ds_str.h, 102
- ds_str_clear
 - ds_str.c, 92
 - ds_str.h, 102
- ds_str_compare
 - ds_str.c, 93
 - ds_str.h, 102
- ds_str_compare_cstr
 - ds_str.c, 93
 - ds_str.h, 103
- ds_str_concat
 - ds_str.c, 93
 - ds_str.h, 103
- ds_str_concat_cstr
 - ds_str.c, 93
 - ds_str.h, 103
- ds_str_create
 - ds_str.c, 94

- [ds_str.h](#), [103](#)
- [ds_str_create_direct](#)
 - [ds_str.c](#), [94](#)
 - [ds_str.h](#), [104](#)
- [ds_str_create_sprintf](#)
 - [ds_str.c](#), [94](#)
 - [ds_str.h](#), [104](#)
- [ds_str_cstr](#)
 - [ds_str.c](#), [94](#)
 - [ds_str.h](#), [104](#)
- [ds_str_decorate](#)
 - [ds_str.c](#), [95](#)
 - [ds_str.h](#), [105](#)
- [ds_str_destroy](#)
 - [ds_str.c](#), [95](#)
 - [ds_str.h](#), [105](#)
- [ds_str_destructor](#)
 - [ds_str.c](#), [95](#)
 - [ds_str.h](#), [105](#)
- [ds_str_doubleval](#)
 - [ds_str.c](#), [95](#)
 - [ds_str.h](#), [105](#)
- [ds_str_dup](#)
 - [ds_str.c](#), [96](#)
 - [ds_str.h](#), [105](#)
- [ds_str_getline](#)
 - [ds_str.c](#), [96](#)
 - [ds_str.h](#), [106](#)
- [ds_str_hash](#)
 - [ds_str.c](#), [96](#)
 - [ds_str.h](#), [106](#)
- [ds_str_intval](#)
 - [ds_str.c](#), [96](#)
 - [ds_str.h](#), [106](#)
- [ds_str_is_empty](#)
 - [ds_str.c](#), [97](#)
 - [ds_str.h](#), [106](#)
- [ds_str_length](#)
 - [ds_str.c](#), [97](#)
 - [ds_str.h](#), [107](#)
- [ds_str_size_to_fit](#)
 - [ds_str.c](#), [97](#)
 - [ds_str.h](#), [107](#)
- [ds_str_split](#)
 - [ds_str.c](#), [97](#)
 - [ds_str.h](#), [107](#)
- [ds_str_strchr](#)
 - [ds_str.c](#), [98](#)
 - [ds_str.h](#), [107](#)
- [ds_str_substr_left](#)
 - [ds_str.c](#), [98](#)
 - [ds_str.h](#), [108](#)
- [ds_str_substr_right](#)
 - [ds_str.c](#), [98](#)
 - [ds_str.h](#), [108](#)
- [ds_str_trim](#)
 - [ds_str.c](#), [98](#)
 - [ds_str.h](#), [108](#)
- [ds_str_trim_leading](#)
 - [ds_str.c](#), [98](#)
 - [ds_str.h](#), [108](#)
- [ds_str_trim_trailing](#)
 - [ds_str.c](#), [99](#)
 - [ds_str.h](#), [109](#)
- [ds_str_trunc](#)
 - [ds_str.c](#), [99](#)
 - [ds_str.h](#), [109](#)
- [ds_vector](#), [13](#)
 - [current](#), [14](#)
 - [data](#), [14](#)
 - [data_destructor](#), [14](#)
 - [ds_vector.h](#), [114](#)
 - [free_on_delete](#), [14](#)
 - [size](#), [14](#)
- [ds_vector.c](#)
 - [ds_vector_clear](#), [110](#)
 - [ds_vector_create](#), [110](#)
 - [ds_vector_destroy](#), [111](#)
 - [ds_vector_destructor](#), [111](#)
 - [ds_vector_element](#), [111](#)
 - [ds_vector_get_next_data](#), [111](#)
 - [ds_vector_seek_start](#), [112](#)
 - [ds_vector_set](#), [112](#)
 - [ds_vector_size](#), [112](#)
- [ds_vector.h](#)
 - [ds_vector](#), [114](#)
 - [ds_vector_clear](#), [114](#)
 - [ds_vector_create](#), [114](#)
 - [ds_vector_destroy](#), [115](#)
 - [ds_vector_destructor](#), [115](#)
 - [ds_vector_element](#), [115](#)
 - [ds_vector_get_next_data](#), [115](#)
 - [ds_vector_seek_start](#), [116](#)
 - [ds_vector_set](#), [116](#)
 - [ds_vector_size](#), [116](#)
- [ds_vector_clear](#)
 - [ds_vector.c](#), [110](#)
 - [ds_vector.h](#), [114](#)
- [ds_vector_create](#)
 - [ds_vector.c](#), [110](#)
 - [ds_vector.h](#), [114](#)
- [ds_vector_destroy](#)
 - [ds_vector.c](#), [111](#)
 - [ds_vector.h](#), [115](#)
- [ds_vector_destructor](#)
 - [ds_vector.c](#), [111](#)
 - [ds_vector.h](#), [115](#)
- [ds_vector_element](#)
 - [ds_vector.c](#), [111](#)
 - [ds_vector.h](#), [115](#)
- [ds_vector_get_next_data](#)
 - [ds_vector.c](#), [111](#)
 - [ds_vector.h](#), [115](#)
- [ds_vector_seek_start](#)
 - [ds_vector.c](#), [112](#)
 - [ds_vector.h](#), [116](#)

- ds_vector_set
 - ds_vector.c, [112](#)
 - ds_vector.h, [116](#)
- ds_vector_size
 - ds_vector.c, [112](#)
 - ds_vector.h, [116](#)
- field_lengths
 - ds_recordset, [12](#)
- fields
 - ds_record, [11](#)
- free_on_delete
 - ds_list, [8](#)
 - ds_vector, [14](#)
- get_cmdline_options
 - config.c, [20](#)
 - config.h, [22](#)
- get_configuration
 - config.c, [20](#)
 - config.h, [23](#)
- gl_error_quit
 - gl_errors.c, [126](#)
 - gl_errors.h, [127](#)
- gl_errors.c
 - gl_error_quit, [126](#)
- gl_errors.h
 - gl_error_quit, [127](#)
- gl_log_msg
 - gl_logging.c, [129](#)
 - gl_logging.h, [131](#)
- gl_logging.c
 - gl_log_msg, [129](#)
 - gl_set_logging, [130](#)
- gl_logging.h
 - gl_log_msg, [131](#)
 - gl_set_logging, [131](#)
- gl_set_logging
 - gl_logging.c, [130](#)
 - gl_logging.h, [131](#)
- hash_size
 - ds_map, [10](#)
 - ds_map_str, [10](#)
- head
 - ds_list, [8](#)
- headers
 - ds_recordset, [12](#)
- help
 - params, [16](#)
- hostname
 - params, [17](#)
- key
 - kv_pair_node, [15](#)
- kv_pair_node, [14](#)
 - key, [15](#)
 - next, [15](#)
 - value, [15](#)
- length
 - ds_list, [8](#)
 - ds_str, [13](#)
- lib/database/database.h, [23](#)
- lib/database/db_connection.h, [25](#)
- lib/database/db_entities.c, [26](#)
- lib/database/db_entities.h, [27](#)
- lib/database/db_internal.h, [29](#)
- lib/database/db_query.h, [30](#)
- lib/database/db_reporting.c, [31](#)
- lib/database/db_reporting.h, [33](#)
- lib/database/db_sampledata.c, [34](#)
- lib/database/db_sampledata.h, [35](#)
- lib/database/db_sql.h, [35](#)
- lib/database/db_structure.c, [37](#)
- lib/database/db_structure.h, [39](#)
- lib/database/db_users.c, [40](#)
- lib/database/db_users.h, [42](#)
- lib/database/dummy/db_dummy_create_entities_table_sql.c, [43](#)
- lib/database/dummy/db_dummy_create_users_table_sql.c, [44](#)
- lib/database/dummy/db_dummy_drop_entities_table_sql.c, [44](#)
- lib/database/dummy/db_dummy_drop_users_table_sql.c, [45](#)
- lib/database/dummy/db_dummy_general.c, [46](#)
- lib/database/dummy/db_dummy_list_entities_report_sql.c, [48](#)
- lib/database/dummy/db_dummy_list_users_report_sql.c, [48](#)
- lib/database/mysql/db_mysql_create_entities_table_sql.c, [49](#)
- lib/database/mysql/db_mysql_create_users_table_sql.c, [49](#)
- lib/database/mysql/db_mysql_drop_entities_table_sql.c, [50](#)
- lib/database/mysql/db_mysql_drop_users_table_sql.c, [51](#)
- lib/database/mysql/db_mysql_general.c, [51](#)
- lib/database/mysql/db_mysql_list_entities_report_sql.c, [54](#)
- lib/database/mysql/db_mysql_list_users_report_sql.c, [54](#)
- lib/datastruct/data_structures.h, [55](#)
- lib/datastruct/ds_list.c, [56](#)
- lib/datastruct/ds_list.h, [60](#)
- lib/datastruct/ds_map.c, [65](#)
- lib/datastruct/ds_map.h, [67](#)
- lib/datastruct/ds_map_str.c, [70](#)
- lib/datastruct/ds_map_str.h, [72](#)
- lib/datastruct/ds_record.c, [74](#)
- lib/datastruct/ds_record.h, [78](#)
- lib/datastruct/ds_recordset.c, [82](#)
- lib/datastruct/ds_recordset.h, [86](#)
- lib/datastruct/ds_str.c, [90](#)
- lib/datastruct/ds_str.h, [99](#)
- lib/datastruct/ds_vector.c, [109](#)

- lib/datastruct/ds_vector.h, 113
- lib/file_ops/config_file_read.c, 116
- lib/file_ops/config_file_read.h, 118
- lib/file_ops/delim_file_read.c, 121
- lib/file_ops/delim_file_read.h, 122
- lib/file_ops/file_ops.h, 124
- lib/gl_general/gl_errors.c, 126
- lib/gl_general/gl_errors.h, 127
- lib/gl_general/gl_general.h, 127
- lib/gl_general/gl_logging.c, 128
- lib/gl_general/gl_logging.h, 130
- list_entities
 - params, 17
- list_users
 - params, 17
- lists
 - ds_map, 10
 - ds_map_str, 11
- login
 - main.c, 132
- MAX_BUFFER_SIZE
 - config_file_read.c, 117
- MAX_LINE_SIZE
 - delim_file_read.c, 122
- main
 - main.c, 132
- main.c, 131
 - login, 132
 - main, 132
 - print_help_message, 133
 - print_usage_message, 133
 - print_version_message, 133
 - test_functionality, 133
- main_mss
 - db_mysql_general.c, 53
- next
 - ds_list_element, 9
 - kv_pair_node, 15
- num_fields
 - ds_recordset, 12
- params, 15
 - create, 16
 - database, 16
 - delete_data, 16
 - help, 16
 - hostname, 17
 - list_entities, 17
 - list_users, 17
 - password, 17
 - sample, 17
 - username, 17
 - version, 17
- params_free
 - config.c, 21
 - config.h, 23
- params_init
 - config.c, 21
 - config.h, 23
- password
 - params, 17
- previous
 - ds_list_element, 9
- print_help_message
 - main.c, 133
- print_usage_message
 - main.c, 133
- print_version_message
 - main.c, 133
- records
 - ds_recordset, 12
- sample
 - params, 17
- size
 - ds_vector, 14
- tail
 - ds_list, 8
- test_functionality
 - main.c, 133
- username
 - params, 17
- value
 - kv_pair_node, 15
- version
 - params, 17